Daniel García Sillas

Modelado y Simulación de un Robot Móvil Autónomo Omnidireccional

2012

Universidad Autónoma de Querétaro

Facultad de Informática
Maestría en Ingeniería de Software Distribuido

MODELADO Y SIMULACIÓN DE UN ROBOT MÓVIL AUTÓNOMO OMNI-DIRECCIONAL

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ingeniería de Software Distribuido

Presenta:
Daniel García Sillas

Dirigido por:
Dr. Efrén Gorrostieta Hurtado

Centro Universitario
Querétaro, Qro.
Diciembre del 2012
México

Universidad Autónoma de Querétaro
Facultad de Informática
Maestría en Ingeniería de Software Distribuido

Modelado y Simulación de un Robot Móvil Autónomo Omni-direccional.

TESIS

Que como parte de los requisitos para obtener el diploma/grado de (o la)

Especialidad/Maestro/Doctor en Maestría en Ingeniería de Software Distribuído

Presenta:
Daniel García Sillas

Dirigido por:
Efrén Gorrostieta Hurtado

SINODALES

Dr. Efrén Gorrostieta Hurtado
Presidente — Firma

Dr. José Emilio Vargas Soto
Secretario — Firma

Dr. Saul Tovar Arriaga
Vocal — Firma

Dr. Juan Manuel Ramos Arreguín
Suplente — Firma

Dr. Jesús Carlos Pedraza Ortega
Suplente — Firma

M.C. Ruth Angelica Rico Hernández
Director de la Facultad

Dr. Irineo Torres Pacheco
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Diciembre del 2012
México

RESUMEN

Aunque la robótica ha avanzado hasta el punto de ser relativamente accesible con bajos presupuestos, aún cabe la necesidad de crear modelos que representen fielmente el comportamiento de un robot con mayor rapidez y bajo este esquema se puede lograr haciéndolo de manera virtual, esto nos permite probar de manera inmediata algoritmos de comportamiento, de inteligencia artificial como control por lógica difusa o redes neuronales, etc. La presente tesis aborda el tema sobre el modelado y simulación de robots móviles autónomos, particularmente hablando de un robot omnidireccional con ruedas. Se plantea el desarrollo de un modelo cinemático así como la implementación utilizando diversas herramientas reconocidas, hasta llegar al punto de la utilización de software especializado en el ramo de la simulación de robots. Durante el desarrollo se evaluaron diversas metodologías y sobre todo el trabajo se basó en lograr un simulador, el cual nos permitió habilitar un siguiente trabajo sobre el comportamiento que debería asumir la arquitectura del robot. El presente trabajo nos permitió desarrollar conocimientos y habilidades en el área de la cinemática, robótica y en el uso de herramientas de software muy específicas.

(**Palabras clave**: robots móviles, robótica, cinemática, modelación y simulation)

SUMMARY

Although robotics has made an incredible progress to the point of being relatively accessible low budgets projects, it is still the need to create models that accurately represent the behavior of a robot faster and under this scheme can be achieved by making it virtually, this allows us to test immediately behavior algorithms, artificial intelligence and fuzzy logic control and neural networks, etc. This thesis addresses the topic on modeling and simulation of autonomous mobile robots, particularly talking about a robot with omnidirectional wheels. It proposes the development of a kinematic model and the implementation using various tools recognized to the point of using specialized software in the field of simulation of robots. During the development of different methodologies were evaluated and all work is based on achieving a simulator, which allowed us to enable a subsequent work on the action it should take the architecture of the robot. This work allowed us to develop knowledge and skills in the areas of kinematics, robotics and the use of very specific software tools.

(**Key words**: mobile robots, robotics, kinematics, modeling and simulation)

A mis padres,
por enseñarme a trabajar y continuar en esta lucha.

A mis maestros,
por aportar sus conocimientos en mi formación.

A mi director de tesis,
por su guía en el presente trabajo.

A mi Jeni,
por alimentar mi espíritu con su amor y paciencia.

AGRADECIMIENTOS

La presente Tesis es un esfuerzo en el cual, directa o indirectamente, participaron varias personas leyendo, opinando, corrigiendo, teniéndome paciencia, dando ánimo, acompañando en los momentos de crisis y en los momentos de felicidad.
Agradezco al Dr. Efrén Gorrostieta por haber apoyado desde involucrarnos en tema un no del todo desconocido pero que no se había incursionado en el hasta ahora, el apoyo y dirección que mantuvo todo el tiempo del desarrollo de este trabajo. A los compañeros de la facultad, al Dr. Carlos Pedraza por su incondicional opinión sobre el presente

Gracias también a mis queridos compañeros, que me apoyaron y me permitieron entrar en su vida durante estos años de convivir dentro y fuera del salón de clase. Que no menciono nombres por no dejar fuera a alguno.

A mis hermanos, a mi madre y a mi padre que me han apoyado y dado aliento incondicionalmente y que a pesar de la distancia que nos separa siempre supe que estuvieron allí. A ti Bebe, que desde un principio hasta el día hoy sigues dándome ánimo en esta aventura llamada maestría, por escucharme, y por tu paciente compañía en toda esta travesía.

Gracias a todos.

I N D I C E

Página

Resumen……..i
Summary…….....ii
Dedicatorias…….iii
Agradecimientos.....iv
Índice.....v
Índice de tablas.....vi
Índice de figuras.....vii
1. INTRODUCCION.....1
2. MARCO TEORICO 6
2.1. Descripción del modelo del robot.....6
2.2. Cinemática.....8
2.3. Herramientas de diseño, modelación y simulación.....10

3. METODOLOGÍA.....20
3.1. Descripción del modelo del robot…….....20
3.2. Modelo tridimensional del robot.....24
3.3. Modelo cinemático.....37
3.4. Simulación del robot en MatLab.....46
3.5. Modelado y simulación en Marilou.....54

4. RESULTADOS Y DISCUSION.....78
4.1. Resultados generales.....78
4.2. Trabajo futuro.....79

5. REFERENCIAS BIBLIOGRÁFICAS.....81

APENDICE.....82
A.1. Código Fuente de la Interface de Control en Visual C#.....84

INDICE DE TABLAS

Tabla **Página**

3.2.1 Listado de objetos que forman el objeto rueda en el modelo del robot..... 32
3.3.1 Parámetros iniciales del modelo cinemático..... 42
3.5.1 Relación de cuerpos geométricos y rígidos..... 61

3.5.2 Relación de articulaciones contra cuerpos rígidos y ejes..... 66

INDICE DE FIGURAS

Figura **Página**

1.1 Robot manipulador industrial..... 1
1.2 AIBO, el perro robot..... 2
1.3 Robots en diferentes formas y aplicaciones..... 2
2.1.1 Estructura del robot triangular..... 6
2.3.1 Entorno de trabajo de 3D Studio MAX R3..... 12
2.3.2 Entorno de trabajo de MATLAB..... 15
2.3.3 Entorno de trabajo de Marilou..... 18
3.1.1 Modelo del robot triangular..... 21
3.1.2 Rueda sueca..... 22
3.1.13 Ejes de movimiento y rotación..... 23
3.2.1 Dibujo del cuerpo del robot en 3DStudio..... 25
3.2.2 Dibujo de la primera extremidad..... 26
3.2.3 Modificación de la altura de uno de los extremos..... 27
3.2.4 Copia y reducción de la extremidad..... 27
3.2.5 Empalme de los dos objetos..... 28
3.2.6 Modificación booleana del objeto principal de la extremidad..... 29
30
3.2.7 Modelo final de la extremidad..... 30
3.2.8 Cuerpo del robot con extremidades sin ruedas.. 31
3.2.9 Modelo de una rueda del robot.....
3.2.10 Conjunto de objetos que forman la rueda del robot..... 31
3.2.11 Modelo completo del robot omnidireccional en 3D Studio..... 33
34
3.2.12 Objetos agrupados de la rueda “whellXX”.....
3.2.13 Objetos agrupados de la rueda y su base “wheelbaseXX”..... 34
35
3.2.13 Objetos agrupados de la pata “legbaseXX”..... 37
3.3.1 Esquema de la estructura cinemática genérica. 41
3.3.2 Esquema cinemático del robot omnidireccional
3.4.1 Diálogo de 3D Studio para exportación de archivos..... 47
47
3.4.2 Menú “File” en Matlab.....
3.4.3 Mundo virtual del robot omnidireccional en Matlab..... 49

Figura **Página**

3.4.4 Secuencia del movimiento circular del robot..... 53
3.4.5 Secuencia del movimiento lineal del robot..... 53
3.5.1 Diálogo de configuración del ambiente en Marilou..... 55
3.5.2 Objetos para construir el físico del robot..... 57
3.5.3 Cuerpo del robot y menú de propiedades del objeto..... 58
3.5.4 Parámetros de la pata del robot..... 59
3.5.5 Cuerpo triangular y tres patas del robot..... 59
3.5.6 Diseño del robot omnidireccional completo en Marilou..... 60
3.5.7 Asignación de cuerpo rígido a los objetos geométricos..... 60
62
3.5.8 Panel de articulaciones..... 63

3.5.9 “Hinge2” colocado en la rueda..... 64
3.5.10 “Hinge” colocado entre la pata y el cuerpo del robot..... 65
65
3.5.11 Cuerpos rígidos y articulaciones del robot..... 66
3.5.12 Panel de 3D..... 67
3.5.13 Zona de sensado de la pata 1..... 69
3.5.14 Panel Modify/Devices..... 70
3.5.15 Mundo virtual y robot omnidireccional..... 71
3.5.16 Simulación y verificación del proyecto..... 73
3.5.17 Diálogo “Device Explorer”..... 74
3.5.18 Interface de control del robot..... 75
3.5.19 Conexión con el robot..... 76
3.5.20 prueba de movimiento del robot..... 77
3.5.21 Prueba de dirección del robot.....
3.5.22 Prueba de rotación de las patas y las ruedas..

1. INTRODUCCION

Existe un creciente auge en el área de la ingeniería electrónica y de sistemas. Algunos de los temas que en ocasiones estas disciplinas comparten es la automatización y la robótica.

Dentro de la automatización de procesos industriales y ya sea en el área doméstica y educativa, se puede observar que cada vez mas intervienen los sistemas robóticos realizando tareas tanto sencillas como de gran complejidad. Algunas de estas tareas pueden ser como aquellas aplicadas en la industria para ejecutar tareas repetitivas tales como la aplicación de soldadura en una línea de ensamble de autos, ordenamiento de objetos, montaje de piezas, etc. En este tipo de tareas se pueden encontrar los que se conocen como robots manipuladores (Figura 1.1), aunque también puede encontrarse robots móviles empleados para transportar objetos del área de producción a una bodega.

Figura 1.1. Robot manipulador industrial.

En el área doméstica pueden encontrarse algunos casos como robots con fines recreativos donde la mayoría de ellos son robots móviles, algunos casos son como las mascotas y robots que ejecutan tareas domésticas, que al final como menciona Ollero (2001) todos ellos aportan un bien y un servicio a nuestra sociedad.

Figura 1.2. AIBO, el perro robot.

Dentro de la robótica existen básicamente los robots manipuladores que usualmente son fijos en una base y los móviles que pueden ser con ruedas o con patas. Estos últimos han experimentado un alto desarrollo durante los últimos años, y a los cuales se les ha encontrado una amplia gama de aplicaciones en diferentes disciplinas, de igual manera, la morfología que llegan a tomar puede ser muy interesante ya que existen desde aplicaciones terrestres, submarinas, anfibias y aéreas, tal es el caso del estudio realizado por Rui Ding et Al (2009) en el cual exponen el modelado cinemático y simulación de un robot anfibio.

Figura 1.3. Robots en diferentes formas y aplicaciones.

Sin importar el tipo de robot, esta disciplina permite el desarrollo de diferentes temas como el modelado cinemático, el modelado dinámico, el desarrollo del control que implica el diseño y escritura de software. Esto motiva la investigación y desarrollo, dando como resultado un considerable número de documentos sobre el tema.

La presente tesis es producto del interés sobre la robótica y particularmente en el modelado cinemático de un robot móvil. Apunta al desarrollo de una metodología para la implementación de un simulador para un robot móvil con

ruedas, y a su vez de establecer un entorno para futuras investigaciones sobre el desarrollo de algoritmos de inteligencia artificial para el control de robots.

Definición del Proyecto.

El tema de la presente tesis habla sobre el desarrollo de un simulador de un robot móvil en particular, con la finalidad de crear una plataforma para futuros desarrollos en el área de control de robots. La intención principal es la de crear el modelo de un robot móvil y poder implementar dicho modelo en un aplicación de software y poder ver gráficamente la estructura física del robot y poder manipularlo desde una interface de software y que permita también la integración con otros sistemas donde puedan desarrollarse algoritmos de control.

Justificación.

La robótica es un tema que actualmente llama la atención de investigadores donde puedan desarrollar técnicas de control de robots entre otras cosas, pero debido a la gran velocidad con que avanza la tecnología algunas implementaciones de robots pueden llegar a tomar mucho tiempo en concretarse, por lo que una manera prácticamente más rápida y viable es la utilización de simuladores, donde pueden realizarse las tareas de desarrollo del control de los robots sin tener que implementarlo físicamente, esto ayuda dándole mayor rapidez a la investigación y no tener que invertir en componentes, equipos y herramientas para la construcción del modelo físico. De tal suerte, que con el presente trabajo queremos dar una herramienta para futuras investigaciones y desarrollos en robótica y a su vez de crear un entorno práctico para el diseño de otros modelos de robots.

Objetivo.

Un objetivo primordial que se describe aquí es la inmediata aplicación de las asignaturas que a lo largo del curso se han tomado, por mencionar algunas es la aplicación de los sistemas distribuidos y la aplicación de lenguajes de alto nivel orientados a objetos. Por otro lado, sin lugar a duda es la de desarrollar las habilidades en ámbitos no tan conocidos como es el tema de la robótica que podría decirse que no es la principal materia de nuestra formación, lo cual le da un tinte mas intenso.

Se quiere también contribuir con un sistema que propiamente pueda ser utilizado para apoyar en la investigación de nuevas tecnologías sobre el tema de robótica, ya que este trabajo ha sido utilizado inmediatamente para el desarrollo de otros trabajos de investigación, en donde se han aplicado algoritmos de inteligencia artificial como lógica difusa para el control de la dirección del robot, y esto por mencionar como un inicio en el uso del modelo.

Por tanto la propuesta es la de presentar un modelo no solo cinemático sino también gráfico que permita visualizar en una aplicación de software a un robot con las características que se describirán mas adelante y que permita su manipulación como si hubiera sido implementado físicamente. Este es el caso de la metodología utilizada por Nory Afzan (2007), en donde utiliza Workspace para la modelación de un robot manipulador.

Alcance.

Dentro de los resultados que se pretenden lograr es la de definir el tipo de robot, en este caso la propuesta es sobre un robot con tres ruedas que se mueve horizontalmente y con nueve grados de libertad. Para ello, es necesario lograr el modelo gráfico y entender sus características físicas. De allí, se pretende

desarrollar el modelo cinemático dado el caso y en función de la plataforma que se pretende utilizar, ya que existen algunas herramientas que operan de acuerdo al modelo cinemático del robot y otras que no es tan necesario y ofrecen gran versatilidad en la implementación de robots lo cual se verá en el transcurso de la presente.

Finalmente, se desea que el sistema quede preparado para que pueda integrarse con sistemas de software y así poder desarrollar algoritmos de control sobre el robot. Entonces, esto nos lleva a tener una cuidadosa selección en las herramientas a utilizar y quede el sistema abierto a una comunicación con otros y por otro lado que las herramientas no sean algo oculto o que tal vez sean complicadas de usar.

2. MARCO TEORICO.

En este capítulo se describe el modelo del robot que se pretende desarrollar, con la intención de acotar el siguiente trabajo y de dar una mejor perspectiva del proyecto. Otros puntos que se exponen son la descripción de algunas herramientas de dibujo asistido por computadora utilizado para obtener un modelo tri-dimensional del robot, así como algunas aplicaciones que permiten realizar la simulación del modelo. Enseguida se hace una breve descripción del concepto de cinemática.

2.1. Descripción del modelo del robot.

El modelo del robot que se pretende desarrollar está basado en una plataforma móvil omnidireccional con ruedas. El cuerpo del robot se describe como triangular y en cada uno de sus vértices está articulado, en cada articulación se encuentra un brazo o pata que a su vez en el extremo opuesta tiene una eje en donde esta la rueda.

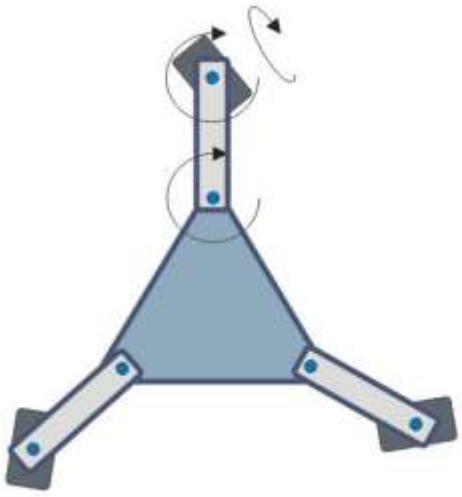


Figura 2.1.1. Estructura del robot triangular.

Dentro de los requerimientos del robot se establece que este debe ser autónomo completamente, debe tener nueve grados de libertad, tres por cada pata consistiendo en la articulación que se encuentra ligada al cuerpo siguiendo por el eje que permite girar la rueda y finalmente la tracción de las rueda para su desplazamiento.

Todos los movimientos del robot deben ser bidireccionales, es decir, que se debe permitir el giro de las ruedas hacia adelante y hacia atrás de forma independiente, las patas deben poder girar en ambos sentidos también.

En cuanto a las dimensiones, estas no están especificadas debido a que solo se quiere obtener un modelo para simulación de los movimientos y no para la construcción física, aunque para la construcción del modelo en la computadora se tomen algunas dimensiones probables para el modelo físico.

Dentro de la literatura encontrada en función del mismo tema, se encontraron algunos parecidos como el modelo propuesto por V.F. Muñoz Martínez et al (2010), el cual es un cuerpo circular y considera tres brazos dispuestos en tres posiciones equidistantes que llegan a formar un triángulo equilátero. En el trabajo de V.F. Muñoz Martínez et al, se presenta el modelado cinemático y el dinámico el cual es bastante cercano al que se propone en el presente trabajo. Ishikawa (2010) presenta el desarrollo y experimento de un robot de tres patas también aunque en este modelo no se incluye el grado de libertad que permite girar a la rueda y cambiar de dirección, aunque se logra ver el desarrollo del modelo cinemático del cuerpo y de las primeras articulaciones.

Para los casos anteriores, se puede encontrar la información sobre el modelo cinemático y en algunos casos el modelo dinámico de robots móviles omnidireccionales pero en varios no se ilustra la implementación en algún medio. En el caso de V.F. Muñoz Martínez et al (2010) demuestra el desarrollo del

modelo de forma gráfica al mostrar el desplazamiento y el comportamiento del robot haciendo una simulación por computadora. Es importante, lograr una representación gráfica para llevar a cabo no simplemente la demostración del modelo si no también para producir un sistema que permite futuros desarrollos en otras áreas como la del desarrollo de técnicas de control e implementación de algoritmos de inteligencia artificial aplicados a los robots, por supuesto sin dejar de ver los conceptos fundamentales de la modelación como es la cinemática.

2.2. Cinemática.

La cinemática es la ciencia del movimiento que trata el tema sin considerar las fuerzas que lo ocasionan. Dentro de esta ciencia se estudian la posición, la velocidad, la aceleración y todas las demás derivadas de alto orden de las variables de posición (con respecto al tiempo o a cualquier otra variable). En consecuencia, el estudio de la cinemática de robots se refiere a todas las propiedades geométricas y basadas en el tiempo del movimiento. Las relaciones entre estos movimientos y las fuerzas y momentos de torsión que los ocasionan constituyen el problema de la dinámica. Es el estudio más básico de como se comporta un sistema mecánico. En robótica móvil, se necesita entender el comportamiento mecánico del robot para diseñar apropiadamente robots móviles para ciertas tareas y para entender como crear el software de control para una instancia de hardware de un robot móvil.

Por supuesto, los robots móviles no son los únicos sistemas mecánicos complejos que requieren tal análisis. Los robots manipuladores han sido objeto de un estudio intensivo por más de tres décadas. De hecho, en algunos casos los robots manipuladores pueden llegar a ser más complejos que algunos móviles (Bravo, 2009): un robot soldador puede tener de cinco o más articulaciones, mientras que algunos robots móviles más antiguos pueden ser solo máquinas con movimiento controlado por solo ruedas. En los recientes años, la comunidad

robótica ha logrado un entendimiento bastante amplio sobre la cinemática e incluso la dinámica (relacionado a masa y fuerza) de robots manipuladores. Por ejemplo, el espacio de trabajo de un manipulador es crucial porque este define el rango de posiciones que puede ser alcanzado. El espacio de trabajo de un robot móvil es igualmente importante porque este define el rango de posibles poses que el robot pudiera alcanzar en su ambiente. El control de un brazo robótico define la forma en la cual la participación de los motores se usa para mover de posición a posición en el espacio de trabajo. Similarmente, el control de un robot móvil define las posibles trayectorias y rutas en el espacio de trabajo. La dinámica de los robots agrega restricciones adicionales al espacio de trabajo debido a las consideraciones de fuerza y masa.

La gran diferencia entre un robot móvil y un brazo manipulador agrega un reto importante en la estimación de la posición. Un manipulador tiene un final fijo en el ambiente. El cálculo de la posición de la carrera final de un brazo es cuestión de entender la cinemática del robot y la medición de todas las posiciones de las articulaciones intermedias. La posición del manipulador es entonces calculable viendo la información del sensor de posición actual. Pero un robot móvil es un autómata que puede moverse completamente por si mismo en su ambiente. No existe un camino directo para medir la posición instantánea de un robot móvil. En lugar de eso, se debe integrar el movimiento del robot sobre el tiempo. Además, se pueden agregar errores al cálculo de la estimación del movimiento sobre algunas variables como derrape y está claro que el cálculo preciso de la posición del robot es un reto bastante desafiante. Existen dos técnicas fundamentales dentro de la cinemática de un robot:

Cinemática Directa. Consiste en determinar la posición y orientación del extremo final del robot con respecto al sistema de la base del robot o a partir de conocer los valores de las articulaciones y los parámetros geométricos. En la cinemática directa se conoce la longitud de cada pieza, el ángulo en cada articulación y la velocidad, y lo que se busca es la posición final del objeto.

Cinemática Inversa. Resuelve la configuración que debe adoptar el robot para una posición y orientación conocidas al extremo. En cinemática inversa se conoce la longitud o dimensión de las piezas y la posición a la que se desea llegar y se busca el ángulo de cada articulación necesitados para obtener la posición así como su velocidad.

2.3. Herramientas de diseño, modelación y simulación.

Existe una gran variedad de herramientas para el diseño, algunos de ellos dedicados a la modelación de objetos en general y algunos otros enfocados a aspectos de ingeniería, pero en general la mayoría aporta recursos para el diseño que permiten la elaboración de un modelo en particular y propiamente hablando de un modelo de robot que se desee representar.

Estas herramientas son aplicaciones de software que permiten realizar el diseño de algún objeto en tres dimensiones dándole mucho realismo y permite a su vez obtener una perspectiva más clara de lo que se pretende. Una de estas herramientas es 3D Studio. Por otro lado, existen programas que permiten manipular los gráficos u objetos creados y aplicarle cierto dinamismo de tal suerte que se pueda representar el comportamiento de dichos objetos, en este caso hablamos de representar la cinemática o el estudio del movimiento de un robot creado por computadora en tercera dimensión al cual se le puede aplicar un modelo cinemático y simular el movimiento real que tendría el robot en el espacio. Este tipo de tareas puede ejecutarse en programas tales como Matlab, el cual es un software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio.

Cabe mencionar que el estudio de la cinemática no incluye el estudio de fuerzas, torque ni propiedades de la masa de los objetos, para ello existen otras

herramientas que permiten no solo la simulación o la representación del movimiento, si no también la simulación de la dinámica lo que significa un estudio mas detallado y apegado a un modelo real. Se puede decir también, que en algunos casos se ha llegado primero a la realización de un robot físico y después se ha desarrollado el modelo cinemático como el caso que menciona He Bin et al (2010), en donde lleva a cabo la construcción de un robot de dos ruedas autobalanceable y enseguida obtiene el modelo para su estudio posterior.

3D Studio.

Hoy en día, se está cada vez más acostumbrado a ver imágenes generadas por computadora en la televisión o en el cine, e incluso en revistas y periódicos. El campo del diseño gráfico ha pasado de ser un reducto para expertos informáticos a ser una profesión normal que a mucha gente le gustaría ejercer. Este programa permite la creación de gráficos en tercera dimensión así como la animación de los gráficos desarrollado por Autodesk. La primera versión comercial de la aplicación fue en 1990 y fue creada por el grupo Yost y fue construida para DOS. Este programa es conocido actualmente como 3D Studio MAX y es una de los programas de animación 3D más utilizados, especialmente para la creación de video juegos, anuncios de televisión y películas.

Las primeras versiones de 3D Studio fueron hechas para correr en MS-DOS, pero en el 1996 el programa fue rescrito y lanzado para correr en Windows NT para obtener lo que hoy se conoce como 3D Studio MAX. El nombre actual de la aplicación es Autodesk 3ds Max y la última versión disponible en el mercado es la 2012 la cual puede correr en versiones mas nuevas de Windows además de poderse instalar en 32 o 64 bits.

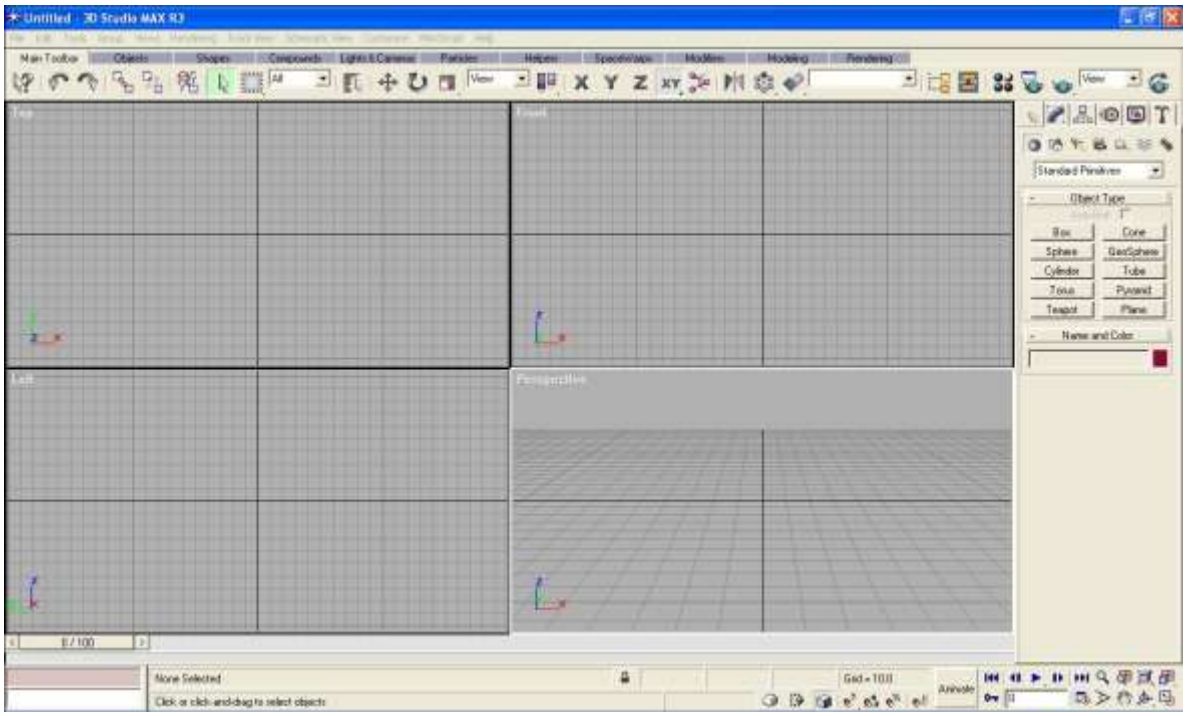


Figura 2.3.1. Entorno de trabajo de 3D Studio MAX R3.

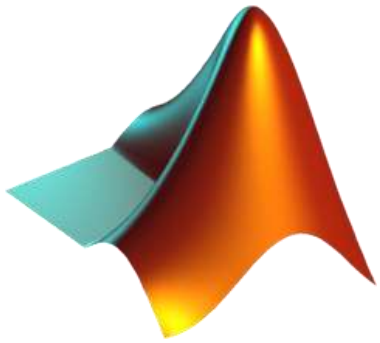
Cuando se habla de 3D significa que se está trabajando en tres dimensiones, es decir: anchura, profundidad y altura. Al dar un vistazo por una habitación todo lo que vemos es tridimensional: la silla, la mesa, las paredes, etc. Pero cuando hablamos de un gráfico tridimensional generado por computadora, el uso del término 3D no es completamente fiel a la verdad. En realidad, estos gráficos tridimensionales son una representación en dos dimensiones de un mundo virtual en tres dimensiones.

Para comprenderse mejor, imaginemos que estamos filmando la habitación con una cámara de video. Al movernos por la habitación, encontraremos varios objetos en tres dimensiones, pero cuando reproduzcamos la cinta en el video, lo que vemos es una imagen plana, en dos dimensiones que representa al mundo en tres dimensiones que acabamos de filmar. La escena tiene realismo gracias a las luces, colores y sombras que parecen dotar a la escena de vida y de una profundidad tridimensional, aunque en realidad es de dos dimensiones.

En los gráficos creados por computadora, los objetos sólo existen en la memoria de la computadora. No tienen una forma física; sólo son fórmulas matemáticas. Como los objetos no existen fuera de la computadora, la única manera de grabarlos es añadir más fórmulas para representar las luces y las cámaras.

En muchos aspectos, trabajar con un programa como 3D Studio MAX es muy parecido a filmar una habitación llena de objetos construidos por uno mismo. El programa permite diseñar la habitación y lo que contiene mediante el uso de varios objetos tridimensionales tales como cubos, esferas, cilindros y conos. Después de haber creado y colocado los objetos en la escena podemos elegir darles cierta textura como plástico, madera o piedra. Después de esto, se pueden agregar cámaras para grabar una vista de ella.

MATLAB.



Este es un software matemático que ofrece un entorno de desarrollo integrado con un lenguaje de programación propio y es una de las muchas herramientas sofisticadas de computación disponibles en el mercado para resolver problemas matemáticos. MATLAB es una abreviatura de MAtrix LABoratory, que traducido es Laboratorio de Matrices.

MATLAB se especializa en el manejo de matrices y es ampliamente utilizado en universidades e institutos de aprendizaje en cursos básicos y avanzados de matemáticas, ciencias y especialmente ingeniería. En la industria se utiliza habitualmente en investigación, desarrollo y diseño de prototipos. El programa estándar de MATLAB comprende una serie de herramientas que pueden ser utilizadas para resolver problemas comunes. Pero MATLAB incorpora, además, otras librerías llamadas toolboxes, que son colecciones de funciones especializadas y diseñadas para resolver problemas muy específicos. Como ejemplos de estas colecciones se podrían citar las ideadas para el procesamiento de señales, el cálculo simbólico y el diseño de sistemas de control.

Hasta hace poco, la mayoría de los usuarios de MATLAB eran persona que tenían conocimientos sobre lenguajes de programación como FORTRAN o C, y que decidieron cambiarse a MATLAB una vez que este software se hizo suficientemente popular. En consecuencia, la mayor parte de la literatura disponible sobre MATLAB supone de una u otra forma que el lector posee conocimientos sobre programación. En los últimos años, sin embargo, MATLAB se enseña en los institutos como el primer lenguaje de programación que aprenden los estudiantes. Aunque en muchas clases de ingeniería, la realización de cálculos con un programa de computación matemático como MATLAB sustituye la

programación de computadoras más tradicional, pero no significa que el usuario no deba aprender un lenguaje de alto nivel como C++ o cualquier otro.

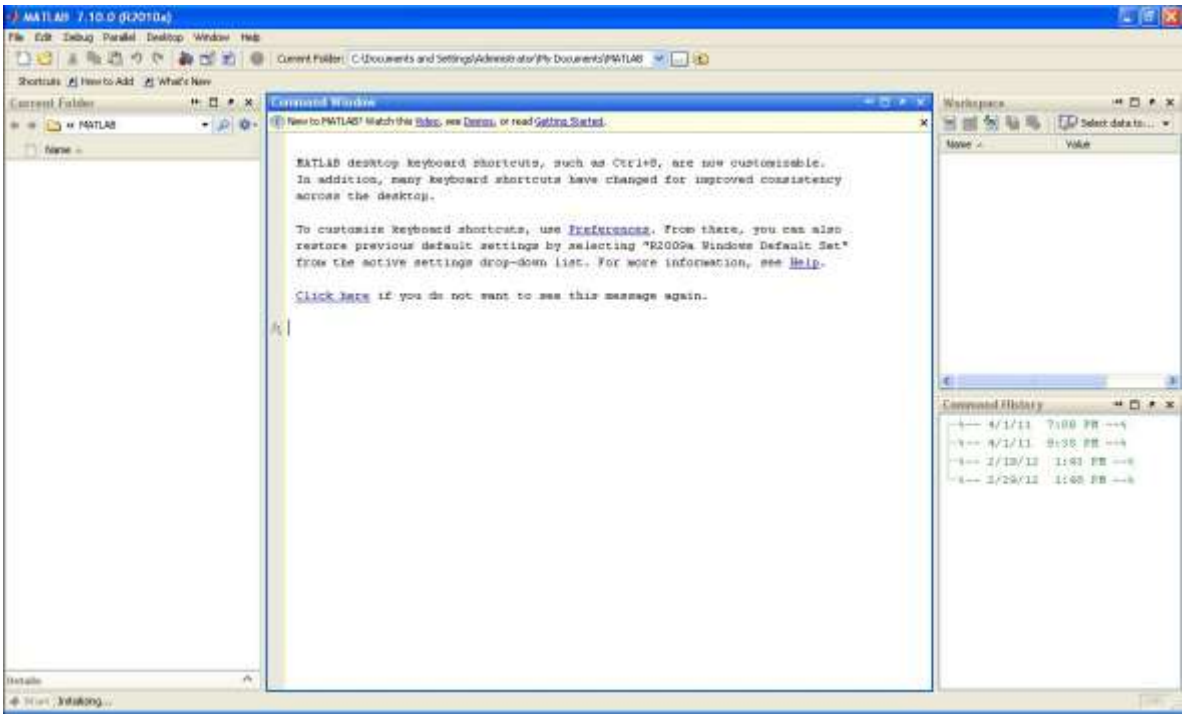


Figura 2.3.2. Entorno de trabajo de MATLAB.

En MATLAB, pueden definirse funciones propias y programas que son almacenados en archivos con la extensión *.m, estas funciones y programas pueden agruparse para realizar un determinado tipo de cálculo. Éste es el origen del concepto de Toolbox (Holly Moore 2007). Existen versiones de MATLAB para una gran variedad de plataformas y sistemas operativos, dentro de ellos Microsoft Windows, Mac OSX y Linux. De la misma manera se dispone de versiones profesional y estudiantil, las cuales son muy similares.

En años recientes se han desarrollado estudios sobre métodos de control para máquinas complejas como robots pero todos estos estudios requieren de mucho tiempo, así que, con el uso de herramientas como Matlab se reducen considerablemente los tiempos empleados en el desarrollo de estos modelos (Yuan Shaoqiang, et al 2008).

MARILOU.

Marilou es una herramienta de simulación que provee de un ambiente completo de modelación y simulación de escenas, así como de robots sujetos a las leyes de la física. Marilou ofrece una amplia gama de posibilidades diseñado para permitir la creación rápida de modelos.

Fue diseñado para ayudar a acelerar proyectos de desarrollo de robótica. Es prácticamente un motor que reproduce con un alto nivel de realidad con respecto a las propiedades en un entorno físico. Permite diseñar escenarios complejos, crear y probar algoritmos de flotas enteras de robots sin tener que invertir en hardware o lidiar con recursos limitados. Así mismo, permite reutilizar los modelos creados en otros escenarios.

Marilou entonces es una aplicación desarrollada para investigación y educación en el área de la robótica, aplicando completamente conceptos de física a los modelos y escenarios creados. Es una forma de experimentar con aplicaciones de todo tipo, incluyendo industriales, comerciales o educativas. Aporta una herramienta para probar aplicaciones en situaciones que pueden ser difícil o tal vez imposibles de reproducir en la realidad. Se pueden probar algoritmos bajo distintos escenarios y configuraciones.

Los robots son máquinas costosas y complejas que a veces son muy difíciles de programar. Teniendo en cuenta el número infinito de situaciones diferentes de un robot puede encontrar (las situaciones que los desarrolladores no siempre eran capaces de planificar con anticipación, o que la inteligencia artificial no es capaz de manejar), así como el riesgo de dañar en realidad un robot durante la fase de desarrollo (errores irreversibles, tales como una caída desde una gran altura), es a menudo, mejor el uso de un robot virtual en un mundo virtual. Los errores cometidos en un entorno virtual durante la fase de desarrollo, por definición, no son de gran trascendencia. Por lo tanto, los simuladores de robótica, ofrecen ventajas significativas.

Marilou es un simulador que se refiere específicamente a las necesidades de los especialistas en robótica. Esto incluye ser capaz de representar fenómenos del mundo real con el fin de llevar a cabo posteriormente experimentos virtuales que hacen posible el estudio de movimiento y la reacción con el tiempo. En Marilou, el simulador permite a los desarrolladores crear los entornos y los robots, permitiendo a las pruebas de comportamiento que se llevarán a cabo en diversas situaciones.

Por ejemplo, considérese el caso de una plataforma de robots a cargo de elaborar las paletas. Con el fin de mantener los costos operativos de abajo, el equipo de producción pide a la robótica a cargo de la plataforma robótica para optimizar la velocidad de cada tarea. El ingeniero por lo tanto debe encontrar una trayectoria que satisface la petición, mientras que al mismo tiempo, el robot es realmente sujeto a las restricciones mecánicas y estructurales y otros factores ambientales, tales como colisiones.

En este caso, es mejor para la robótica utilizar un simulador que hará que sea posible poner a prueba caminos o trayectorias, con la posibilidad de cometer errores sin dañar realmente cualquier equipo real. Después de eliminados los problemas, la trayectoria puede ser validada para ejecutarlo en el robot real.

Sin embargo, es importante recordar que, para que el simulador sea eficiente, el modelo que se use debe ser funcional y estructuralmente correcto. Es decir, deben ser construidos de modo que los resultados obtenidos y las predicciones del simulador sean relevantes para la tarea que pretenden llevar a cabo.

Marilou ayuda robótica en cada paso del camino en este complejo proceso. Marilou ofrece lo último en tecnologías de simulación robótica, y utiliza el motor de renombre Open Dynamics (ODE) con el fin de ofrecer a los usuarios todas las

herramientas que necesitan para simular proyectos de robótica. Interfaz gráfica intuitiva Marilou le ayuda a realizar su trabajo rápidamente

Marilou le guiará a través de las diversas tareas necesarias para la simulación robótica adecuada. Los párrafos siguientes proporcionan una visión general rápida de las principales partes involucradas en el uso de la aplicación.

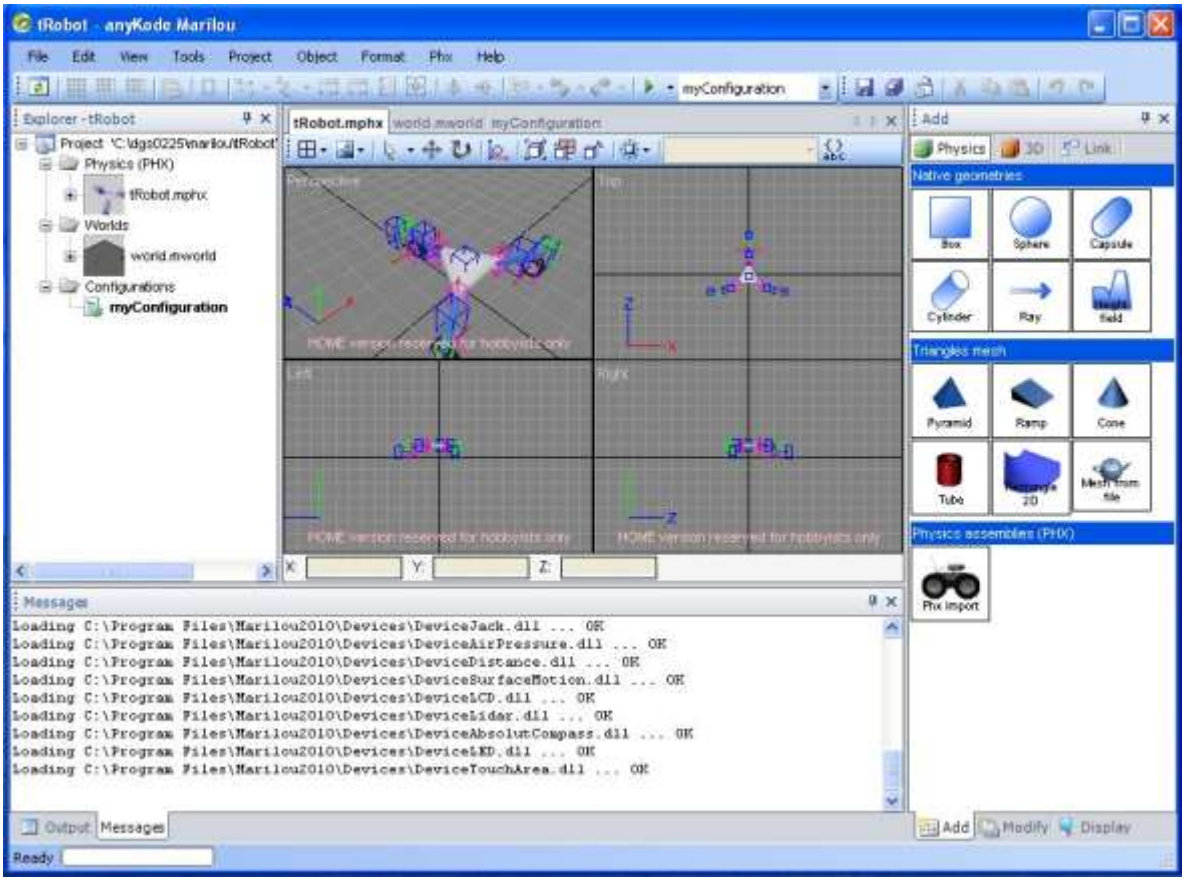


Figura 2.3.3. Entorno de trabajo de Marilou.

Algunas de las características de Marilou son:

- Manejo completo de robots en forma gráfica así como de los modelos de los ambientes.
- Múltiples vistas.

- Manejo de cuerpos rígidos.
- Restricciones mecánicas.
- Propiedades de superficies.

Marilou por su parte también soporta múltiples lenguajes de programación como C, C++, C#, J#, y puede correr bajo Windows y/o Linux. Es compatible con Matlab y Java, y contiene documentación en diferentes idiomas.

En lo que se refiere a la simulación, Marilou contiene componentes que aportan realismo a los modelos creados, tales como son sensores, motores, servo motores, odómetros, medidores o sensores de distancia, radares laser, detectores de choque, actuadores cilíndricos, cámaras, GPS, acelerómetros/giroscopio, brújulas y mas.

Marilou fue creado por ANYKODE, quien cuenta con más de quince años de experiencia en las ciencias de la computación y robótica, quien desarrolla y comercializa la aplicación.

3. METODOLOGIA

La demanda en el desarrollo de prototipos robóticos a un bajo coste y cada vez más rápido apunta a la aplicación de este tipo de herramientas como son la modelación y simulación. Además como se mencionó, con la ventaja de poder ejecutar simulaciones bajo condiciones extremas como es el comportamiento impredecible de algunos dispositivos físicos que expongan de cierta manera la seguridad e integridad del o los usuarios. Por otro lado, se puede llevar a cabo simulaciones de ambientes peligrosos, de equipos grandes sin siquiera tener que realizar una inversión en la instalación de dichas máquinas y tampoco considerar consumos de energía, así como la utilización de un número importante de dispositivos actuadores como motores, dispositivos neumáticos, sensores, etc. Lo que nos deja además de poder seleccionar los tipos mas adecuados para la aplicación que se desee desarrollar.

Dado a estas características en el uso de simuladores se puede pensar en un diseño particular que se ajuste a las necesidades del ambiente, pudiendo hacer pruebas también sobre algún modelo dedicado a ciertas aplicaciones.

3.1. Descripción del modelo del robot.

De acuerdo a la necesidad de desarrollar un prototipo especial propio para una aplicación se trabajó en el presente robot que cumple con una morfología omnidireccional, lo que significa, que el robot cuenta con múltiples grados de libertad que le da un grado muy alto de maniobra.

Dentro de los tipos de robots tenemos robots móviles o fijos, y a su vez dentro de los móviles están aquellos que pueden desplazarse con ruedas o por medio de extremidades articuladas.

El modelo utilizado se basa en un robot móvil omnidireccional con ruedas y que a su vez tiene tres extremidades para darle dirección y forma al robot. El robot tiene una forma triangular y en cada uno de los vértices tiene un brazo o extremidad articulada y a su vez cada una de las extremidades tiene un eje vertical donde se encuentra cada una de las ruedas del robot, y donde cada una de las tres ruedas incluidas deben ser direccionables.

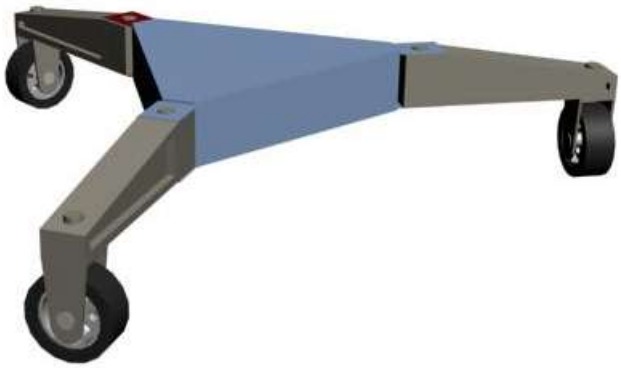


Figura 3.1.1. Modelo del robot triangular.

Las extremidades y así como los ejes de dirección de las ruedas deben ser controladas por medio de servo motores o en este caso deben cumplir con la funcionalidad de un servo motor de tal suerte que el ángulo pueda ser controlado completamente, de esta forma se controla no solo dirección sino la forma que toma del robot. El robot sigue una morfología basada en el diseño propuesto por Romero Torres et al (2009), donde la dirección de las ruedas pueden girar 360 grados, y el ángulo de las articulaciones está limitado por el espacio que permite el mismo cuerpo del robot, el cual puede ser alrededor de 180 grados.

El desplazamiento de las ruedas está dado por motores convencionales los cuales darían solamente la tracción para poder mover el robot, la condición es también que puedan ser reversibles así el robot pudiera desplazarse hacia adelante o hacia atrás sin tener que hacer movimientos adicionales.

Esta rueda es estándar sin características especiales, y en general no se especifica tipo ni calidad de los materiales dado que en el momento se pretende conseguir una representación virtual para para poder ejecutar una simulación. Aunque cabe mencionar que existen diferentes tipos de ruedas como las tipo suecas, que permiten el desplazamiento del robot incluso con las ruedas perpendiculares al movimiento del robot como lo describe Jae-Bok Song et al (2008), la cual deja al robot desplazarse con solo la aplicación de fuerza motriz en algunas ruedas y no todas.

Figura 3.1.2. Rueda sueca.

El estudio de cinemática que se presenta adelante está basado en el uso de una rueda sueca, este tipo de aplicaciones puede llegar a reducir la complejidad en el sistema de control de movimiento del robot, es decir, las aplicaciones de software pueden simplificarse debido a que solo se controla la dirección general del robot y no de las ruedas en si.

Por otro lado, para el objetivo de esta tesis no es necesario especificar las dimensiones del robot, solo es importante la forma y las especificaciones funcionales. Aunque, se dieron dimensiones para poder llevar a cabo el diseño, que se especifican más adelante.

La siguiente figura muestra los grados de libertad del robot. Las flechas rectas de color amarillo indican que es un eje, y cada uno de los ejes pueden rotar en ambas direcciones.

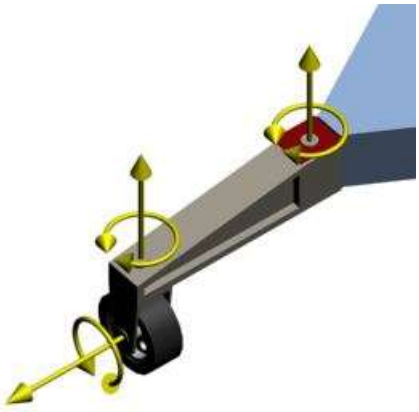


Figura 3.1.3. Ejes de movimiento y rotación.

Cada uno de los ejes indicados así como sus rotaciones aplica para cada una de las tres extremidades que conforman el robot triangular. Como puede observarse, todas las articulaciones o grados de libertad se definen bidireccionales incluyendo la del eje de traslación de las ruedas.

En total se describen nueve grados de libertad, tres por cada extremidad, lo que hace que el robot adquiera cierta complejidad para el sistema de control que pueda ser implementado para esta configuración.

En términos generales en el modelo propuesto así como en la simulación no se toma en cuenta tampoco el peso del robot lo que puede llegar a afectar en la modelación dinámica, pero en nuestro caso el trabajo está centrado en el estudio

cinemático, aunque una de las aplicaciones utilizadas nos permitió incluir características particulares de los materiales como es el caso del peso de cada pieza utilizada en el modelo del robot, aplicando de esta forma mucho más realismo a la simulación.

3.2. Modelo tridimensional del robot.

Con el fin de que el modelo que se obtenga pueda ser utilizado en diferentes ambientes de simulación, es necesario que la herramienta de dibujo del modelo arroje un resultado en un archivo con formato compatible, el cual pueda ser leído por diferentes entornos tal como Matlab. En este caso, la aplicación utilizada para dibujar el modelo en tercera dimensión del robot es 3D Studio.

3D Studio permite dibujar modelos tridimensionales a escala y con gran detalle, y de igual forma permite dar como resultado imágenes en archivos con gran número de formatos. Para el caso práctico, el modelo fue generado en formato vrml (virtual reality modeling language – lenguaje para modelado de realidad virtual), el cual es un formato de archivo normalizado, el cual tiene como objetivo la representación de escenas u objetos interactivos tridimensionales.

El modelo del robot es construido en función de la especificaciones dadas en el punto anterior (descripción del modelo), las cuales realmente no hay una especificación cerrada sobre las dimensiones ni de la forma de cada una de las extremidades. Para el caso práctico del presente trabajo, se requiere solamente de un modelo de dimensiones aproximadas pero que cuente con los grados de libertad especificados.

Dibujo del cuerpo del Robot.

El dibujo del robot inicia con el cuerpo, el cual es definido por un cuerpo sólido denominado en 3D Studio como Gengon de tres lados, con un diámetro de 120, para el caso práctico las unidades definidas son milímetros, un fillet (recorte de esquinas) de 20 y un altura de 40. El objeto Gengon se selecciona desde el menú "**Create**", y luego en "**Extended Primitives**" y enseguida seleccionar "**Gengon**". Es necesario primero dibujarlo arrastrando con el mouse en el área de dibujo y enseguida darle los parámetros de dimensiones y colores. El color en este caso es un azul suave definido en un rango de colores RGB de 154, 185, 229. Dando como resultado el siguiente bosquejo.

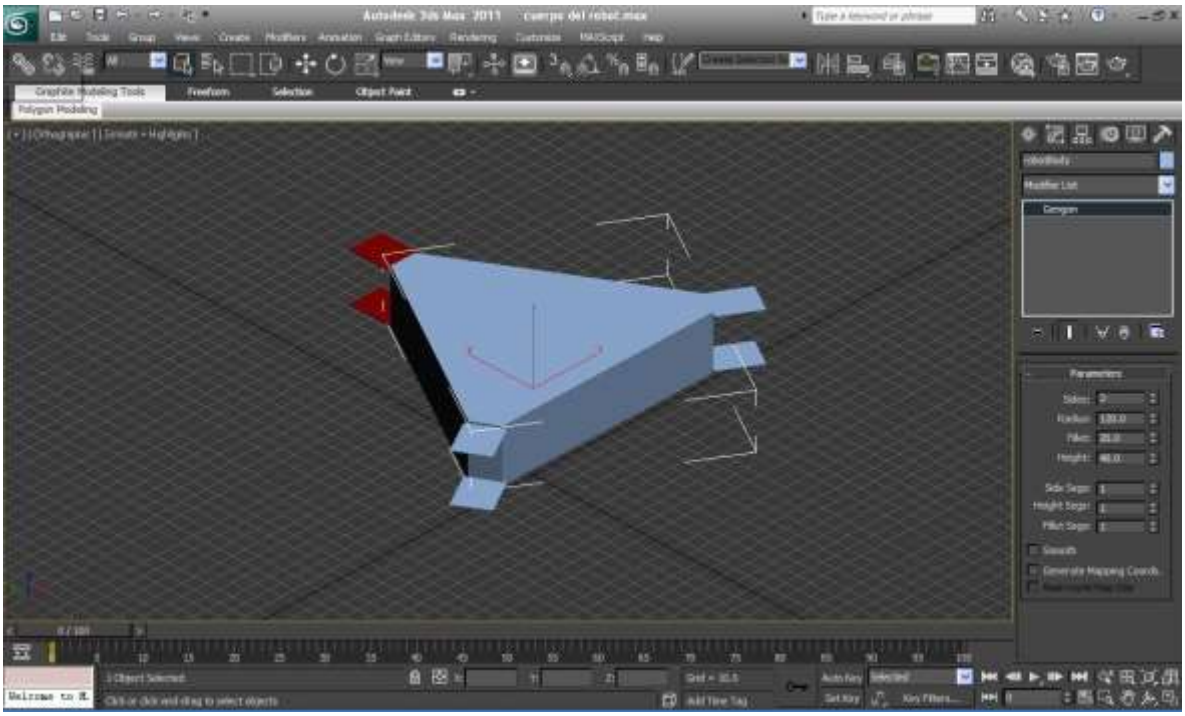


Figura 3.2.1. Dibujo del cuerpo del robot en 3DStudio.

Al mismo cuerpo anexamos un juego rectángulos con dimensiones 25 de ancho y 30 de largo con 2 de alto, como bases de los primeros ejes donde estarán colocadas las extremidades dándole el primer grado de libertad para cada una. Estos rectángulos se dibujan a partir de seleccionar el objeto "**Box**" en el menú "**Create**" y "**Standard Primitives**". Los rectángulos deberán orientarse y alinearse

con el cuerpo Gengon, utilizando las herramientas en el menú "**Select and Rotate**".

Dibujo las extremidades del robot.

La siguiente etapa en el diseño es la creación de las extremidades o patas del robot. Esto se lleva a cabo primero dibujando un rectángulo de 25 de ancho por 120 de largo y 35 de alto. Desde el menú "Create", seleccionar "Standard Primitives" y enseguida "Box". Dibujarlo en el área y enseguida darle las dimensiones en la parte derecha de la aplicación bajo el botón "**Modify**"  .

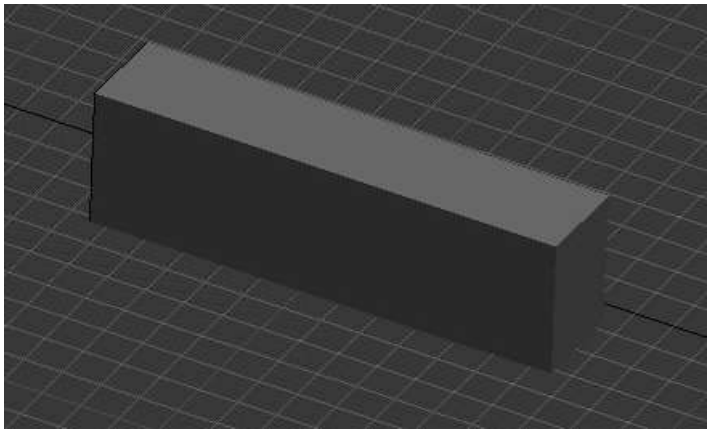


Figura 3.2.2. Dibujo de la primera extremidad.

Para darle una mejor apariencia, se modifica la pieza reduciendo la altura de uno de los extremos. Se selecciona convertir el objeto, haciendo clic con el botón derecho y desde el menú contextual seleccionar "**Convert to editable mesh**", se despliegan alrededor de la figura los puntos que conforman la malla, en este momento se puede seleccionar los vértices seleccionando la opción "Vertex" en la parte derecha de la pantalla  . Enseguida seleccionar los puntos superior extremos de la malla, y bajarlos aproximadamente diez puntos. Se logra una apariencia como la de la Figura 3.2.3 en la página siguiente.

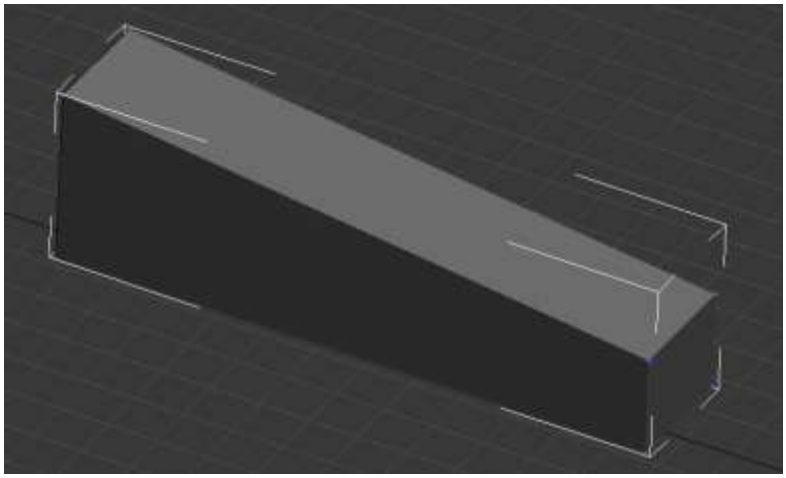


Figura 3.2.3. Modificación de la altura de uno de los extremos.

Una modificación adicional a esta pieza es un entresaque para dar la apariencia de un maquinado y resaltar los bordes. Para ellos es necesario copiar una pieza igual a la de la Figura 3.2.3 y reducirla.

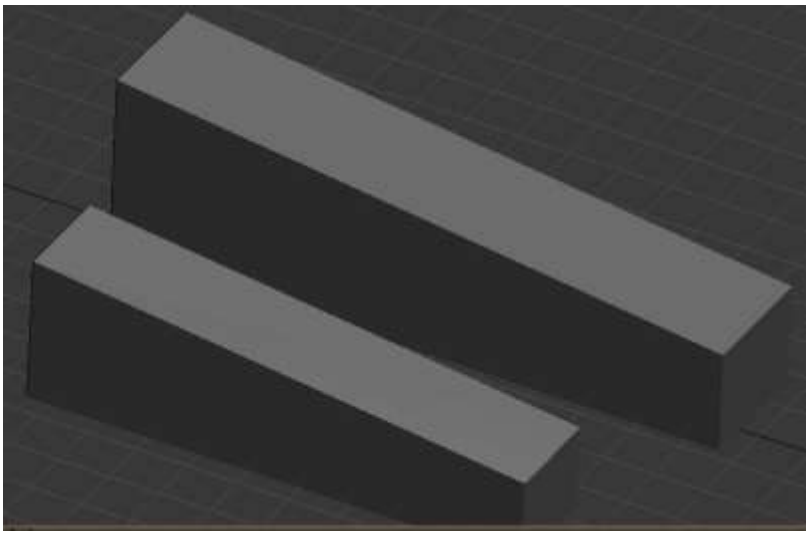


Figura 3.2.4. Copia y reducción de la extremidad.

La intención es obtener una imagen para de extraer una parte de la pieza grande a partir de la pieza pequeña, esta funcionalidad en 3D Studio se le conoce como extracción con operaciones booleanas.

Para ello se mueve la pieza mas pequeña con la herramienta “Select and Move” localizada en el botón en la parte superior de la pantalla por debajo del menú  . La pieza pequeña deberá quedar empotrada en la grande cinco unidades.

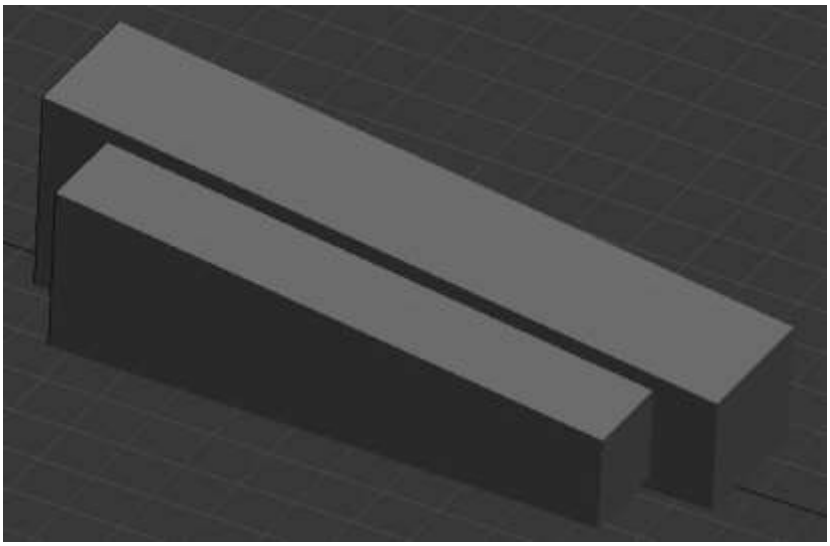


Figura 3.2.5. Empalme de los dos objetos.

Después de colocar juntas las piezas, seleccionar la pieza grande y desde el menú “**Create**” la opción “**Compound**” y “**Boolean**”. En la parte derecha de la pantalla aparecerán las opciones para la función booleana, seleccionar la opción “**Pick Operand B**” y seleccionar la opción “**Reference**” para que se conserve la pieza pequeña, ya que existen otras opciones como “**Move**” que la borrarían del entorno.

La modificación queda como se ilustra en la Figura 3.2.6, se hace la misma operación para el otro lado utilizando la misma pieza, para ello se mueve la pieza

al otro lado con la misma distancia tomada en la primera fase de cinco milímetros y enseguida utilizar la función booleana para extraer la otra parte.

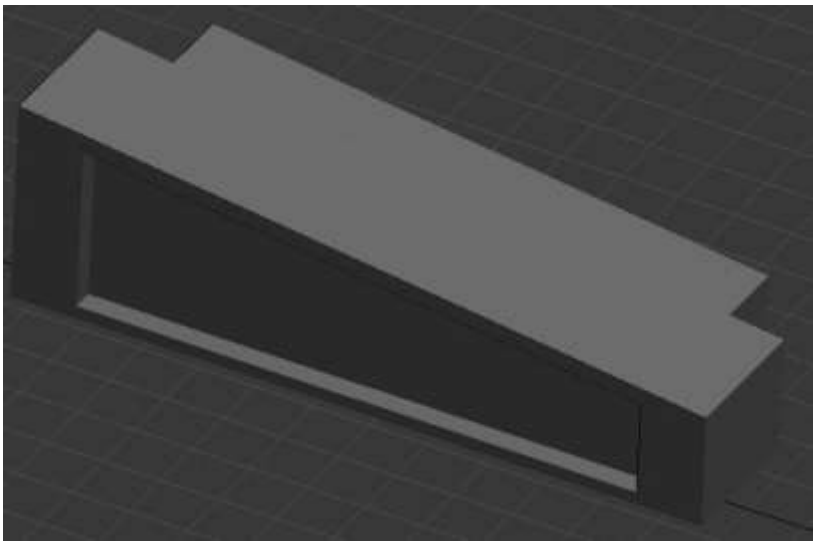


Figura 3.2.6. Modificación booleana al objeto principal de la extremidad.

Finalmente para esta pieza se agrega un par de objetos mas que consiste en un rectángulo "**Box**" y en un cilindro "**Cylinder**", ambos objetos se encuentran en el menú "**Create**" bajo "**Standard Primitives**". El rectángulo queda con dimensiones 25 de ancho y profundo por 35 de alto. El cilindro con 40 de alto y radio 5. Enseguida el cilindro se alinea con el rectángulo con la herramienta "**Align**" bajo el menú "**Tools**" en el centro con los tres ejes y con la opción de alineamiento con el centro. Ambos objetos se coloca en el extremo anterior alto de la extremidad modificada para quedar como en la Figura 3.2.7, donde se ilustra el final de la extremidad completa.

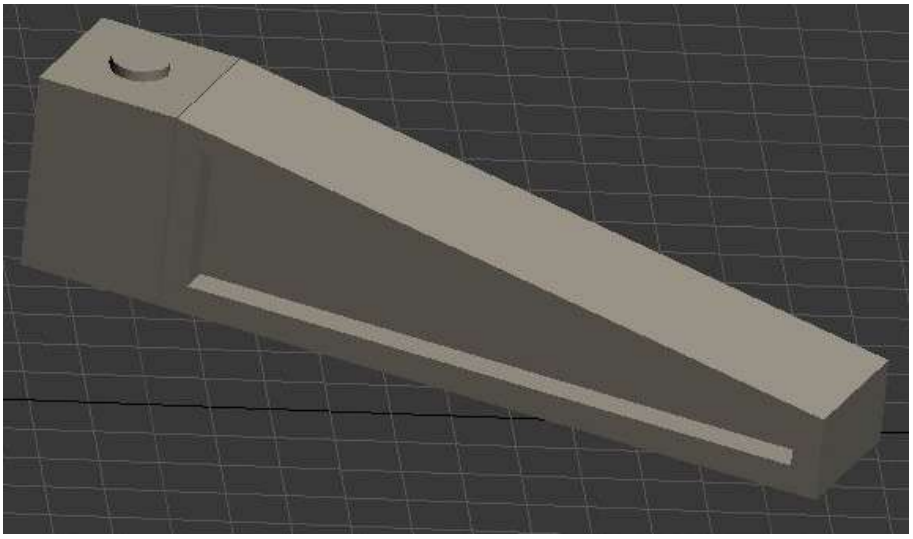


Figura 3.2.7. Modelo final de la extremidad.

El siguiente paso, es la copia de tres objetos de la extremidad recién dibujada para y orientarlos y alinearlos al cuerpo del robot, para lograr un primer acercamiento como el siguiente.

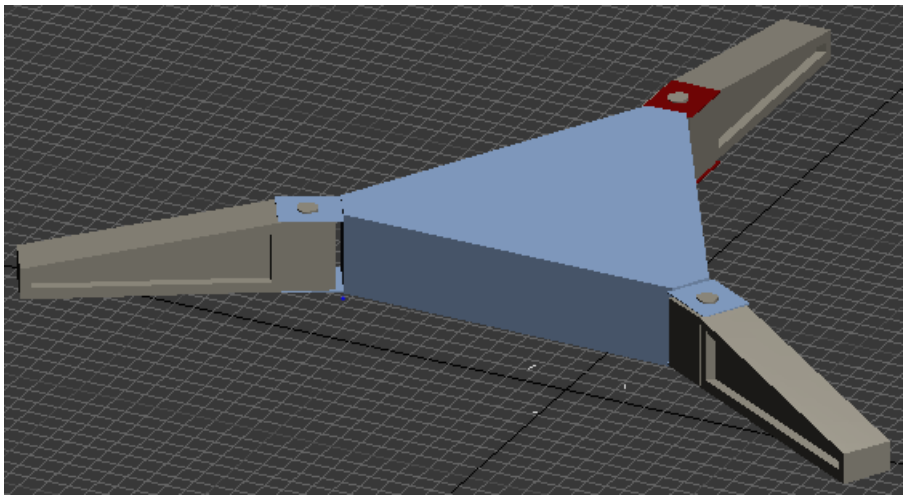


Figura 3.2.8. Cuerpo del robot con extremidades sin ruedas.

Dibujo de las ruedas del robot.

El siguiente paso es la creación de las ruedas del robot (Figura 3.2.9), para obtener también un juego de tres de ellas para completar el modelo del robot. La rueda es un conjunto de objetos varios que forman un objeto completo de llanta, montura de la llanta, el eje de la llanta, el marco y el eje de rotación de la rotación de la rueda (Figura 3.2.10).

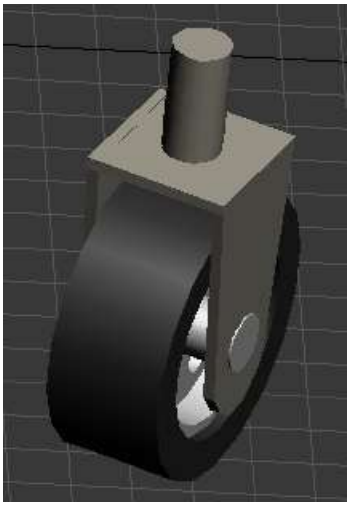


Figura 3.2.9. Modelo de una rueda del robot.

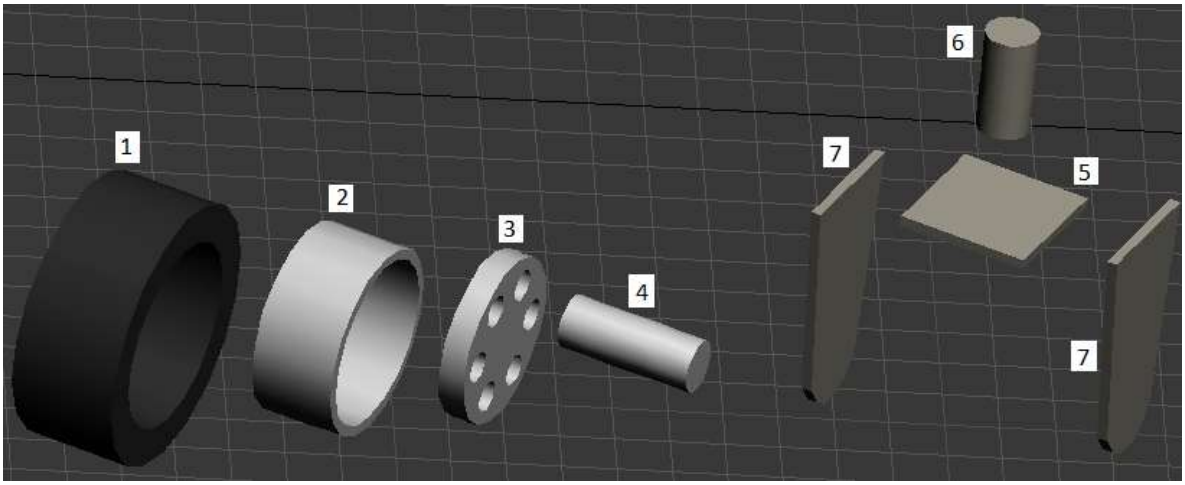


Figura 3.2.10. Conjunto de objetos que forman la rueda del robot.

En la tabla 3.2.1. se encuentra la descripción de los objetos de la rueda.

No.	Objeto	Tipo	Largo	Ancho	Alto	Radio 1	Radio 2	Color	Modificaciones	Descripción
1	Llanta	Cilindro		20		25	18	Negro		Llanta con apariencia de caucho
2	Montura de llanta	Cilindro		16		18	16	Gris plata		Montura exterior
3	Montura de llanta	Cilindro		5		16	0	Gris plata	Seis hoyos a una distancia de 10 a partir del centro, utilizando objetos compuestos booleanos con un cilindro de radio 3.	Montura interior
4	Eje de la montura	Cilindro		26		5	0	Gris		Eje de la llanta
5	Base de la rueda	Caja	25	2	25			Gris		Base horizontal del eje
6	Base de la rueda	Cilindro			20	5		Gris		Eje de rotación la rueda
7	Base de la rueda	Caja	25	2	50			Gris	Modifcacióna los vértices para redondear la parte del fondo	Bases laterales.

Tabla 3.2.1. Listado de objetos que forman el objeto rueda en el modelo del robot.

Una vez que se obtiene una rueda completa se realiza una copia de dos ruedas mas para completar el juego de tres requerido para el modelo completo del robot.

Dibujo final del robot.

Una vez que se dispone de todos los objetos en cuestión descritos para la construcción del robot, se completa el modelo en 3D Studio para quedar con un modelo como el de la Figura 3.2.11, la cual muestra el diseño final del robot y listo para ser implementado en un simulador.

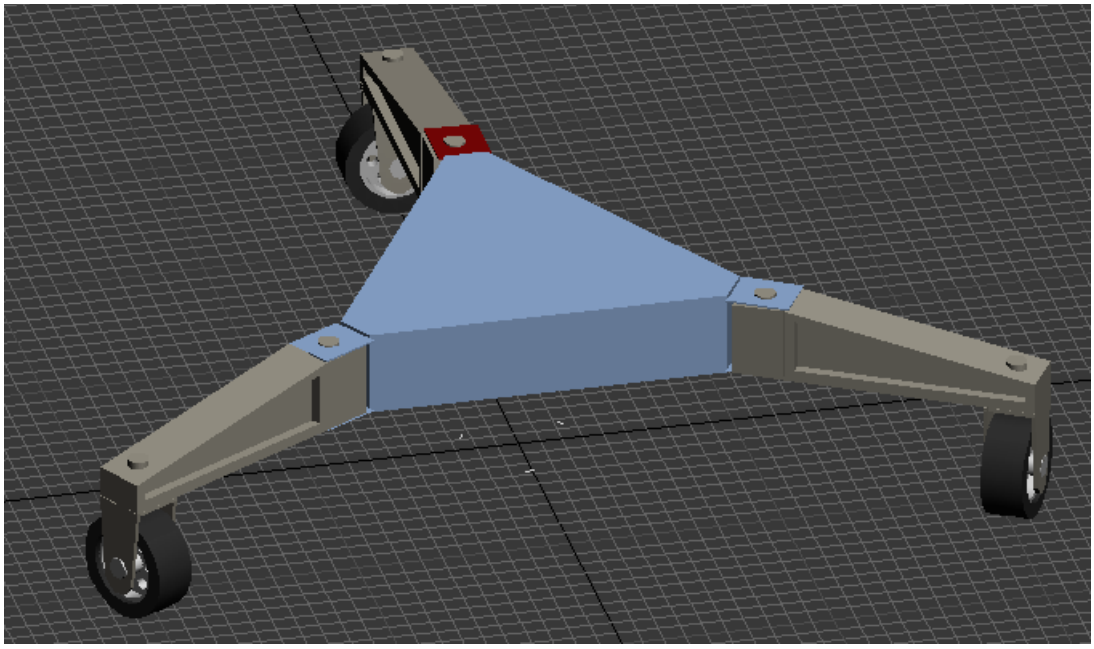


Figura 3.2.11. Modelo completo del robot omnidireccional en 3D Studio.

Agrupamiento y asignación de nombres.

Es importante definir grupos de objetos, con la finalidad de que mas tarde cuando se intente manipular el robot, este se manipule por grupo y no cada objeto creado individualmente. Por ejemplo, la rueda, esta está compuesta por varios objetos, como cilindros, cuadros, etc. Es necesario agrupar todos estos objetos para formar un solo grupo y asignarle un nombre. El nombre asignado a cada grupo de objetos será el que se use mas adelante en la simulación del robot.

Para agrupar, es necesario seleccionar todos los objetos primarios que comprendan un objeto completo para poderlos agrupar. Con el ratón se pueden seleccionar varios objetos haciendo clic en la pantalla y arrastrando para atrapar los objetos en cuestión. Enseguida, desde el menú “Group” seleccionar la opción “Group” de vuelta, aparecerá un dialogo donde se puede teclear el nombre que se le quiera dar al grupo de objetos.

Para este caso se han agrupado y se les han asignado nombres a algunos grupos, tal es el caso de la rueda en donde se agrupan solo los objetos que definen el diseño de la llanta, el nombre que se le asigna es “wheel01”, dado que son tres entonces estos van “wheel02” y “wheel03”.

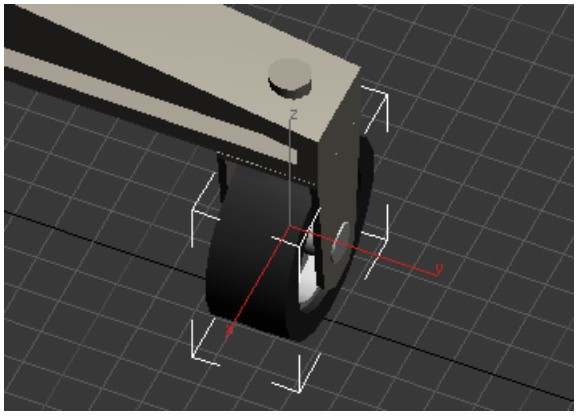


Figura 3.2.12. Objetos agrupados de la rueda “wheelXX”.

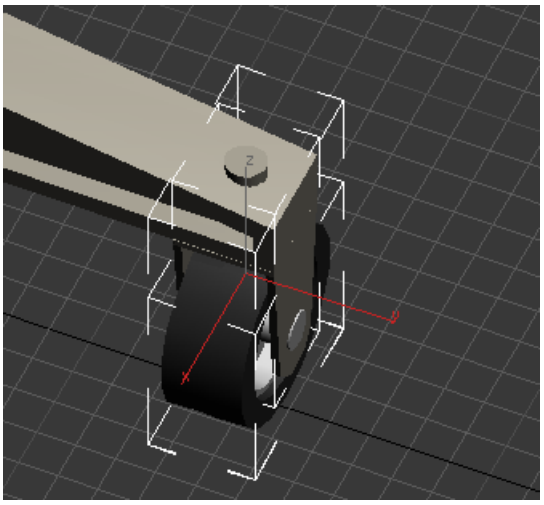


Figura 3.2.13. Objetos agrupados de la rueda y su base “wheelbaseXX”.

Se pueden hacer grupos de varios objetos compuestos, en el caso del robot omnidireccional debido a que existe una conexión entre la rueda y la base de la rueda que tendrá un grado de libertad, esto es, que permitirá a la rueda girar en su propio eje (Figura 3.2.13). El nombre utilizado para el grupo de la pata es “wheelbase01”, del 01 al 03, representado por XX, por ejemplo “wheelbaseXX”.

Es importante definir pequeños grupos en función de los grados de libertad que se quieran lograr, y por otro lado, si se generan muchos grupos se pueden generar también muchos nombres que puede llegar a ser confuso.

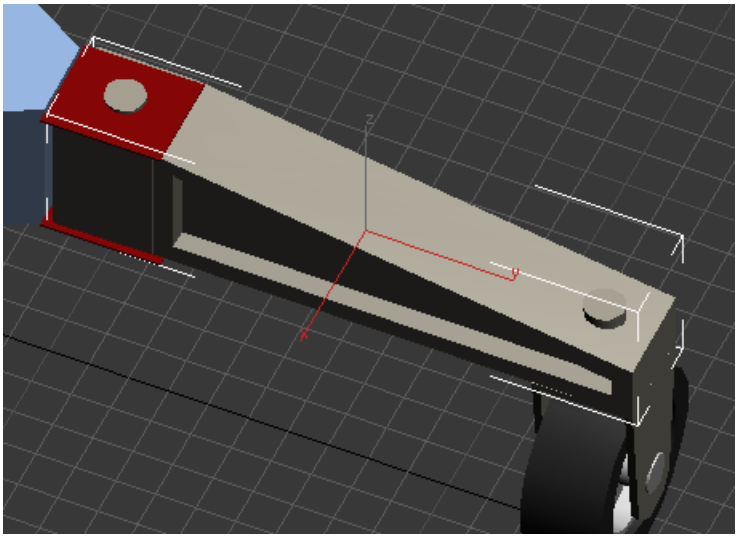


Figura 3.2.13. Objetos agrupados de la pata “legbaseXX”.

La siguiente etapa es agrupar la base de la pata (“legbaseXX”), que está compuesto por los objetos de sujeción al cuerpo del robot. Después el grupo “legbaseXX” se agrupa con el el de “wheelbaseXX”, de esta forma estamos formando el grupo “legXX”, el cual constituyen las extremidades o patas del robot.

Este proceso deberá realizarse para cada una de las patas. Al tener cada grupo de patas “legXX”, estas se podrán agrupar al cuerpo del robot, para formar el último grupo que sería el robot completo y se le denomina “trobot”.

Como se ha descrito previamente el robot omnidireccional tiene programado tener nueve grados de libertad, las cuales están divididas en tres patas dejando tres grados de libertad en cada una. La primera articulación estará controlada para poder girar en forma horizontal con el grupo “legXX”, que debido a que está agrupado al girar en el eje asignado esta desplazará también a la rueda. La siguiente articulación es el giro de la rueda, lo que permitirá darle dirección a la rueda y estará designado por el grupo “wheelbaseXX”. Finalmente, el eje horizontal de la rueda que permitirá el desplazamiento del robot y este está denominado como “wheelXX”. La descripción realizada durante el desarrollo del modelo puede encontrarse en literatura como la Michael Todd Peterson (2000).

La posición global de todo el sistema, incluyendo patas y ruedas esta denominado como “trobot”, el cual comprende el sistema completo de posicionamiento sobre la posición de referencia global.

3.3. Modelo cinemático del robot.

Para realizar el análisis del modelo cinemático del robot omnidireccional, es necesario comenzar describiendo los grados de libertad de los cuales está provisto. Como mencionamos anteriormente, el robot cuenta con un juego de tres extremidades, todas ellas unidas al cuerpo triangular y que a su vez cuentan con tres grados de libertad cada una, para dar un total de nueve.

Cada una de las articulaciones, son bidireccionales y el robot debe ser capaz de trasladarse girando las ruedas en un sentido o en otro. En la Figura 3.3.1, puede observar la estructura cinemática genérica del robot, puede apreciarse los diferentes grados de libertad del robot, además del sistema de elementos que determinan la dirección.

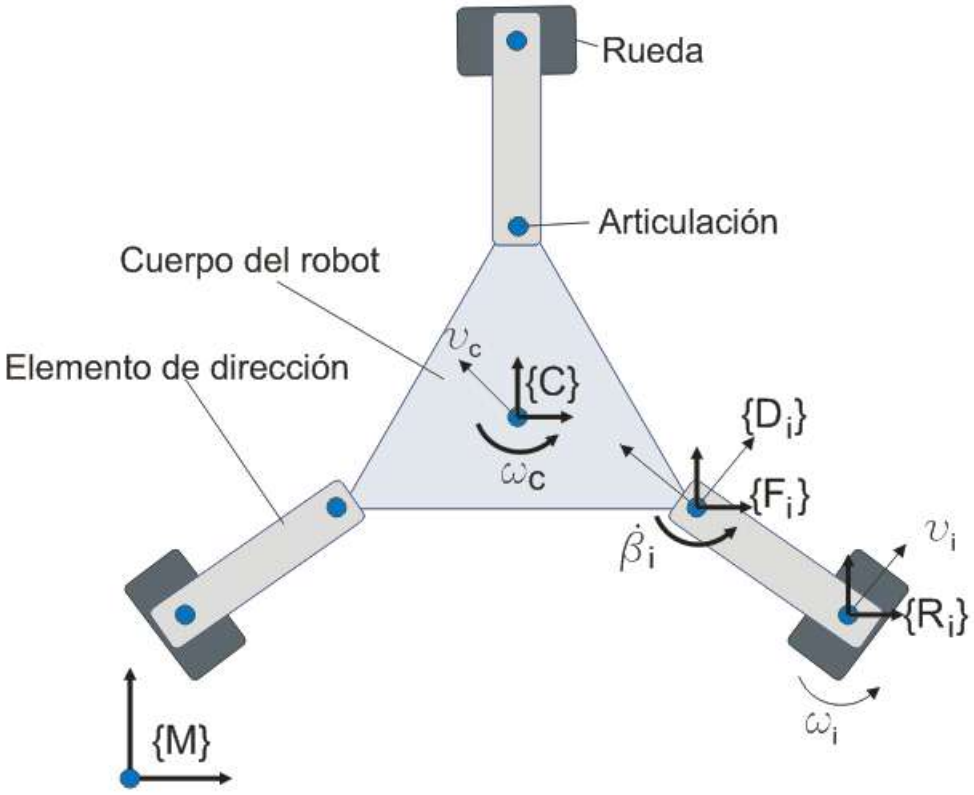


Figura 3.3.1. Esquema de la estructura cinemática genérica.

Este desarrollo está basado en el trabajo realizado por V.F Muños (2010), en donde se desarrolla un modelo cinemático como el que se busca. En la Figura 3.1.15 se ilustran los elementos de dirección o extremidades que se encuentran fijas a las ruedas y a su vez unidas al cuerpo del robot mediante una articulación. Para poder obtener las posiciones y orientaciones de los componentes, se debe asociar un sistema de coordenadas solidario a cada uno, tal y como se describe abajo:

{C}: Asociado al cuerpo del robot y es el punto guía del vehículo, su posición cartesiana es x_c, y_c y su orientación θ_c con respecto a un sistema global de trabajo {M} corresponden a la del robot.

$\{F_i\}$: Es el sistema fijado en el punto de anclaje de la articulación de la extremidad i-ésima. El ángulo α_i representa la orientación relativa de este sistema con respecto al sistema {C}, y su vector de posición es λ_i.

$\{D_i\}$: Referente al elemento de dirección de la rueda i-ésima. El ángulo de dirección entre el sistema actual y el anterior, es β_i. El vector de posición resulta nulo ya que $\{F_i\}$ y $\{D_i\}$ son coincidentes.

$\{R_i\}$: Sistema ubicado en el punto de contacto de la rueda i-ésima con el suelo, tal como aparece en la Figura 3.1.15. El ángulo de dirección, y el vector de posición entre el sistema actual y el anterior son respectivamente Υ_i y δ_i.

La velocidad lineal del robot, entonces estará dada por la ecuación 3.1:

$$v_c = \mathrm{R}(\theta_i.)\ .\ v_i + \omega_i\ .\ p_i + \dot{\beta}_i\ .\ \lambda_i \qquad \text{Ecuación 3.1}$$

Donde R() representa una matriz de rotación en el plano, p_i y θ_i se definen como el vector de posición y la orientación del sistema $\{R_i\}$ visto desde {C], tal y como se indica a continuación:

$$p_i = \lambda_i + R(\alpha_i + \beta_i)\,\delta_i \qquad \text{Ecuación 3.2}$$

$$\theta_i = \alpha_i + \beta_i + \Upsilon_i \qquad \text{Ecuación 3.3}$$

En cuanto a la velocidad angular del robot, sólo se toman en cuanta las velocidades homónimas de la articulación y de la rueda:

$$\omega_i = \dot{\beta}_i - \omega_i \qquad \text{Ecuación 3.4}$$

Las ecuaciones 3.2, 3.3 y 3.4 se organizan en forma de matriz jacobiana, tal y como se indica a continuación:

$$\begin{pmatrix} v_{cx} \\ v_{cy} \\ \omega_c \end{pmatrix} = \begin{pmatrix} c_i & -s_i & p_{iy} & -\lambda_{iy} \\ s_i & c_i & -p_{ix} & \lambda_{ix} \\ 0 & 0 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} v_{ix} \\ v_{iy} \\ \omega_i \\ \beta_i \end{pmatrix} \qquad \text{Ecuación 3.5}$$

$$V_c = \hat{J}_i \cdot \dot{\hat{q}}_i$$

Donde v_{cx} y v_{cy} son las componentes de $v_{\dot{c}}$, p_{ix} y p_{iy} las de p_i, λ_{ix} y λ_{iy} las de λ_i, y por último v_{ix} y v_{iy} las de v_i. Además, c_i y s_i, respectivamente, al coseno y al seno del ángulo θ_i.

La velocidad lineal de la rueda se obtiene a partir del giro de la misma a partir de la acción de un motor. En el caso de una rueda convencional, de tracción y no direccionable, con un radio r_i y una velocidad de giro ω_{ix}, se define la matriz de conversión de la actuación W_i de la siguiente manera:

$$\dot{\hat{q}}_i = W_i \cdot \dot{q}_i = \begin{pmatrix} 0 & 0 \\ -r_i & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \omega_{ix} \\ \omega_i \end{pmatrix} \qquad \text{Ecuación 3.6}$$

De acuerdo a la ecuación 3.6, esta permite introducir la actuación ω_{ix} y anular la acción de dirección debía a $\dot{\beta}_i$. En el supuesto de que la rueda sea direccionable, se empleará la matriz W_i, presentada en la siguiente ecuación:

$$\dot{\hat{q}}_i = W_i \cdot \dot{q}_i = \begin{pmatrix} 0 & 0 & 0 \\ -r_i & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \omega_{ix} \\ \omega_i \\ \dot{\beta}_i \end{pmatrix}$$ Ecuación 3.7

Considerando *n* ruedas en el robot, a partir de la ecuación 3.5, se plantea un sistema de ecuaciones sobre determinado, donde el vector de velocidades V_c tiene que satisfacer simultáneamente las siguientes restricciones:

$$\begin{pmatrix} I \\ I \\ \vdots \\ I \end{pmatrix} \cdot V_c = \begin{pmatrix} J_1 & 0 & \dots & 0 \\ 0 & J_2 & \dots & 0 \\ \dots & \ddots & \ddots & \dots \\ 0 & \dots & 0 & J_N \end{pmatrix} \cdot \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_N \end{pmatrix}$$ Ecuación 3.8

$$A\, V_c = B\, \dot{q}$$

En la ecuación anterior I representa la matriz identidad de tres por tres.

Enseguida, se utiliza una aproximación por mínimos cuadrados con el objeto de encontrar una solución para el vector V_c:

$$V_c = \underbrace{(A^T \cdot A)^{-1} \cdot A^T \cdot B}\, \dot{q}$$ Ecuación 3.9

$$V_c = J\, \dot{q}$$

La matriz J representa entonces el jacobiano completo del robot. A este resultado, se le fija agrega que el error de estimación sea nulo, lo que implica suponer que el robot está actuado de forma apropiada para que no deslice. Entonces se define la función $\Omega(A)$ como:

$$\Omega(A) = A\,(A^T \cdot A)^{-1} \cdot A^T - I$$ Ecuación 3.10

La condición de no deslizamiento se expresa como sigue

$\Omega(A)\ B\ \dot{q} = 0$ Ecuación 3.11

En la siguiente Figura (3.3.2), se ilustra el esquema cinemático que representa la configuración geométrica del robot omnidireccional objeto del estudio, la cual se define por una estructura triangular equilátera, en cuyos vértices están instaladas las ruedas omnidireccionales. La distancia del origen del sistema {C} que corresponde al centro geométrico, a cualquiera de las ruedas viene dada por L. Todas las ruedas se definen como direccionables, y por lo tanto, se produce la igualdad $\{F_c\} = \{D_c\} = \{R_c\}$, es decir para toda i, se cumple $\beta_i = 0^o$ y $\Upsilon_i = 0^o$. La tabla 3.2 enmarca los valores de los parámetros del modelo cinemático.

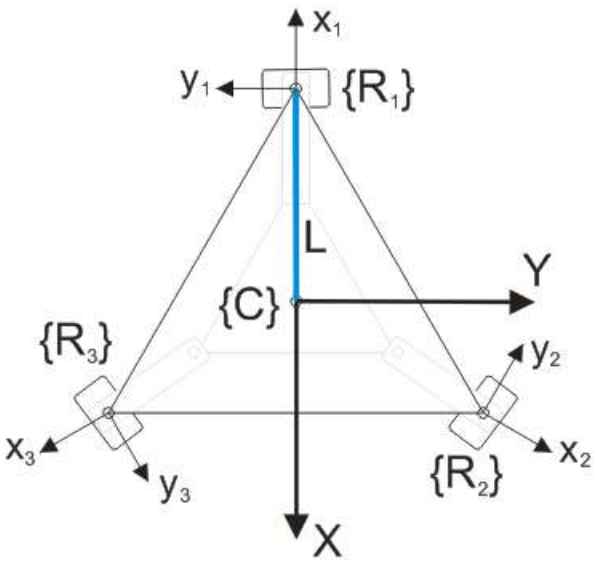


Figura 3.3.2. Esquema cinemático del robot omnidireccional.

	Rueda 1 {R1}	Rueda 2 {R2}	Rueda 3 {R3}
α_i	180^o	60^o	-60^o
β_i	0^o	0^o	0^o
Υ_i	180^o	0^o	0^o
δ_i	(0,0,0)	(0,0,0)	(0,0,0)
λ_i	(-L, 0, 0)	$\left(\frac{L}{2}, \frac{L\sqrt{3}}{2}, 0\right)$	$\left(\frac{L}{2}, \frac{-L\sqrt{3}}{2}, 0\right)$

Tabla 3.3.1. Parámetros iniciales del modelo cinemático.

El objetivo es obtener el jacobiano de la rueda, como lo propone V.F Muños, esto se logra multiplicando la matriz $\hat{J}_i$, que se detalla en la ecuación 3.5, por una matriz de conversión que corresponde a la actuación de las ruedas omnidireccionales, esta se puede ver en la ecuación siguiente (3.12):

$$W_i \,.\, \dot{q}_i = \begin{pmatrix} 0 & r & 0 \\ -R & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} \omega_{ix} \\ \omega_{ir} \\ \omega_{iz} \end{pmatrix} \qquad \text{Ecuación 3.12}$$

La matriz W_i modela una rueda de radio R, omnidireccional con rodillos de radio r a noventa grados, tractora y no direccionable. En referencia al vector $\dot{q}$, ω_{ix} es el grado de actuación del motor, ω_{ir} es la velocidad angular de giro de los rodillos y ω_{iz} el deslizamiento rotacional en el eje vertical de la rueda. De este modo, el jacobiano de la rueda i-ésima queda reflejado de la siguiente manera:

$$J_i = \begin{pmatrix} R\, s_i & r\, c_i & \lambda_i \\ -R\, c_i & r\, s_i & -\lambda_i \\ 0 & 0 & 1 \end{pmatrix} \qquad \text{Ecuación 3.13}$$

Al sustituir los parámetros de la tabla 3.3.1 en la ecuación 3.13, se obtienen los jacobianos de cada rueda:

$$J_1 = \begin{pmatrix} 0 & -r & 0 \\ R & 0 & L \\ 0 & 0 & 1 \end{pmatrix}$$ Ecuación 3.14

$$J_2 = \begin{pmatrix} \frac{R\sqrt{3}}{2} & \frac{r}{2} & \frac{L\sqrt{3}}{2} \\ \frac{-R}{2} & \frac{-r\sqrt{3}}{2} & \frac{-L}{2} \\ 0 & 0 & 1 \end{pmatrix}$$ Ecuación 3.15

$$J_3 = \begin{pmatrix} \frac{-R\sqrt{3}}{2} & \frac{r}{2} & \frac{-L\sqrt{3}}{2} \\ \frac{-R}{2} & \frac{-r\sqrt{3}}{2} & \frac{-L}{2} \\ 0 & 0 & 1 \end{pmatrix}$$ Ecuación 3.16

Las matrices J_1, J_2 y J_3, se componen en la función de la ecuación 3.8, y se resuelve el jacobiano completo del vehículo como se describe en la ecuación 3.9:

$$J = \begin{pmatrix} 0 & \frac{-r}{3} & 0 & \frac{R}{a} & \frac{r}{6} & \frac{L}{a} & \frac{-R}{a} & \frac{r}{6} & \frac{-L}{a} \\ \frac{R}{3} & 0 & \frac{L}{3} & \frac{-R}{6} & \frac{r}{a} & \frac{-L}{6} & \frac{-R}{6} & \frac{-r}{a} & \frac{-L}{6} \\ 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \end{pmatrix}$$ Ecuación 3.17

$$a = 2\sqrt{3}$$

La matriz anterior relaciona la velocidad del vehículo con las de giro que aparecen en las ruedas, pero desde en este caso solo interesan los grados actuados, de tal forma que se impone la condición de no deslizamiento de la ecuación 3.11:

$$\begin{pmatrix} 0 & \frac{2r}{3} & 0 & \frac{R}{a} & \frac{r}{6} & \frac{L}{a} & \frac{-R}{a} & \frac{r}{6} & \frac{-L}{a} \\ \frac{-2R}{3} & 0 & \frac{-2L}{3} & \frac{-R}{6} & \frac{r}{a} & \frac{-L}{6} & \frac{-R}{6} & \frac{-r}{a} & \frac{-L}{6} \\ 0 & 0 & \frac{-2}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 0 & \frac{-r}{3} & 0 & \frac{-R}{\sqrt{3}} & \frac{-r}{3} & \frac{-L}{\sqrt{3}} & \frac{-R}{a} & \frac{r}{6} & \frac{-L}{a} \\ \frac{R}{3} & 0 & \frac{L}{3} & \frac{R}{3} & \frac{-r}{\sqrt{3}} & \frac{L}{3} & \frac{-R}{6} & \frac{-r}{a} & \frac{-L}{6} \\ 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{-2}{3} & 0 & 0 & \frac{1}{3} \\ 0 & \frac{-r}{3} & 0 & \frac{R}{a} & \frac{r}{6} & \frac{L}{a} & \frac{R}{\sqrt{3}} & \frac{-r}{3} & \frac{L}{\sqrt{3}} \\ \frac{R}{3} & 0 & \frac{L}{3} & \frac{-R}{6} & \frac{r}{a} & \frac{-L}{6} & \frac{R}{3} & \frac{r}{\sqrt{3}} & \frac{L}{3} \\ 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{-2}{3} \end{pmatrix} . \dot{q} = 0 \qquad \text{Ecuación 3.18}$$

En la matriz anterior se define como indeterminado debido a que existen tres filas que son combinación lineal de las otras. Dado esto, se despejan las variables no actuadas en función de las que si están, obteniendo el siguiente resultado:

$$\omega_{1r} = \frac{R(\omega_{3x} - \omega_{2x})}{r\sqrt{3}} \qquad \text{Ecuación 3.19}$$

$$\omega_{2r} = \frac{R(\omega_{1x} - \omega_{3x})}{r\sqrt{3}} \qquad \text{Ecuación 3.20}$$

$$\omega_{3r} = \frac{R(\omega_{2x} - \omega_{1x})}{r\sqrt{3}} \qquad \text{Ecuación 3.21}$$

$$\omega_{1z} = \frac{R(\omega_{1x} + \omega_{2x} + \omega_{3x})}{3L} \qquad \text{Ecuación 3.22}$$

$$\omega_{2z} = \omega_{1z} \qquad \text{Ecuación 3.23}$$

$$\omega_{3z} = \omega_{1z} \qquad \text{Ecuación 3.24}$$

Sustituyendo el resultado anterior en la ecuación 3.17, se obtienen las velocidades globales en función de las actuaciones de las ruedas:

$$v_{cx} = \frac{R(\omega_{2x} - \omega_{3x})}{\sqrt{3}} \qquad \text{Ecuación 3.25}$$

$$v_{cy} = \frac{R(2\omega_{2x} - \omega_{2x} - \omega_{3x})}{3} \qquad \text{Ecuación 3.26}$$

$$w_c = -\frac{R(\omega_{1x} - \omega_{2x} - \omega_{3x})}{3L} \qquad \text{Ecuación 3.27}$$

Colocando el resultado en forma matricial se obtiene el jacobiano final actuado del robot móvil:

$$\begin{pmatrix} v_x \\ v_y \\ \omega_c \end{pmatrix} = \begin{pmatrix} 0 & \frac{R}{\sqrt{3}} & \frac{R}{\sqrt{3}} \\ \frac{2R}{3} & \frac{-R}{3} & \frac{-R}{3} \\ \frac{-R}{3L} & \frac{-R}{3L} & \frac{-R}{3L} \end{pmatrix} . \begin{pmatrix} \omega_{1x} \\ \omega_{2x} \\ \omega_{3x} \end{pmatrix} \qquad \text{Ecuación 3.28}$$

La matriz inversa del jacobiano de la ecuación 3.19 queda de la siguiente forma:

$$\begin{pmatrix} \omega_{1x} \\ \omega_{2x} \\ \omega_{3x} \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{R} & \frac{-L}{R} \\ \frac{\sqrt{3}}{2R} & \frac{-1}{2R} & \frac{-L}{R} \\ \frac{\sqrt{3}}{2R} & \frac{-1}{2R} & \frac{-L}{R} \end{pmatrix} . \begin{pmatrix} v_x \\ v_y \\ \omega_c \end{pmatrix} \qquad \text{Ecuación 3.29}$$

3.4. Simulación del robot en Matlab

Una vez obtenidas las piezas fundamentales para realizar una simulación, que son el modelo tridimensional o el diseño del robot en forma digital y el modelo matemático, se puede lograr una simulación para observar el comportamiento del robot en un mundo virtual.

Para llevar a acabo dicha simulación es necesario poner junto toda la información generada en un ambiente que nos permita ejecutar de manera continua el modelo cinemático del robot y representarlo de forma gráfica en el diseño. Una herramienta que nos permite realizar este trabajo es Matlab.

En Matlab la primera tarea es la implementación de la visualización del modelo. Es necesario para ello convertir el modelo creado en 3D Studio en un archivo compatible con VRML 97 (Virtual Reality Model Language – Lenguaje de Modelado de Realidad Virtual), el cual puede ser leído por Matlab y permite una mejor manipulación de los objetos creados en 3DStudio. La conversión se hace desde 3D Studio, exportando el modelo creado desde el menú “File” o “Archivo”, seleccionar la opción “Exportar” y dándole la opción en el menú de VRML97, no es necesario darle parámetros adicionales (Figura 3.4.1). Para el caso práctico se nombró el archivo como “trobot.WRL”.

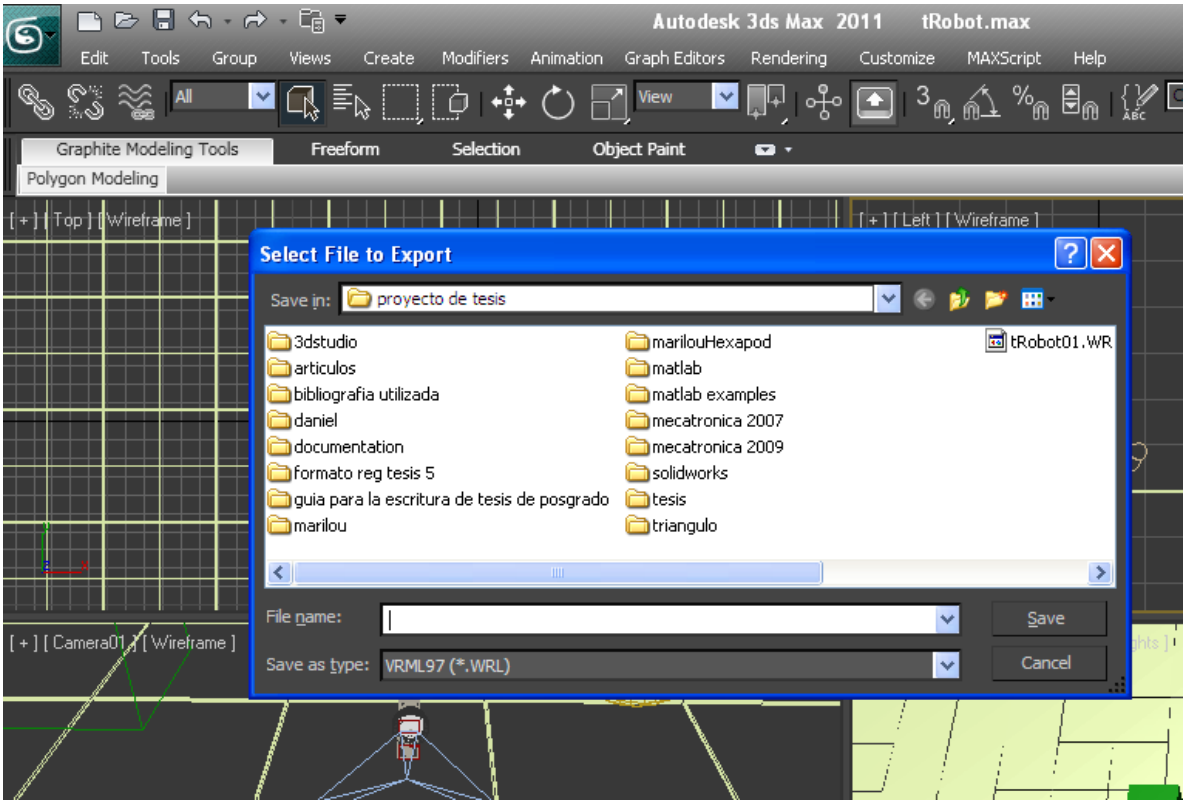


Figura 3.4.1. Diálogo de 3D Studio para exportación de archivos.

Después, es necesario acceder a la aplicación de Matlab y crear un archivo nuevo desde el menú “File”->”New”.

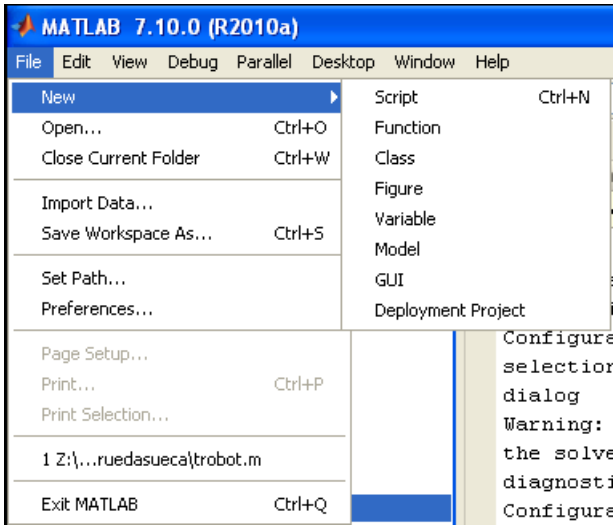


Figura 3.4.2. Menú “File” en Matlab.

Se puede renombrar el archivo, con algún nombre que pueda recordarse, en el presente se denominó como “trobot.m”, la extensión m es dada por Matlab.

Al colocar los archivos “trobot.m” y “trobot.WRL” en un folder se podrán observar en la parte de la izquierda de la aplicación los archivos disponibles para el proyecto en curso.

El paso siguiente es escribir el código necesario para cargar el archivo que contiene todos los objetos gráficos (“trobot.WRL”), y definir todos los nodos que serán manipulados en el proyecto, utilizando las herramientas Simulink®3D Animation ™ de Matlab.

Código en Matlab.

Se inicia haciendo una limpieza en la pantalla actual de Matlab así como de las variables utilizadas.

```
clc;
clear all;
```

Enseguida se crea un objeto de la clase VRWORLD que representa el mundo virtual.

```
world = vrworld('tRobot.wrl');
```

El paso siguiente es abrir el objeto world.

```
open(world);
```

En Simulink este objeto puede verse de dos formas diferentes, con la opción –internal se selecciona el modo por default como visor interno o se puede seleccionar externo para visualizarlo en un navegador web.

```
myview = view(world, '-internal');
vrdrawnow;
```

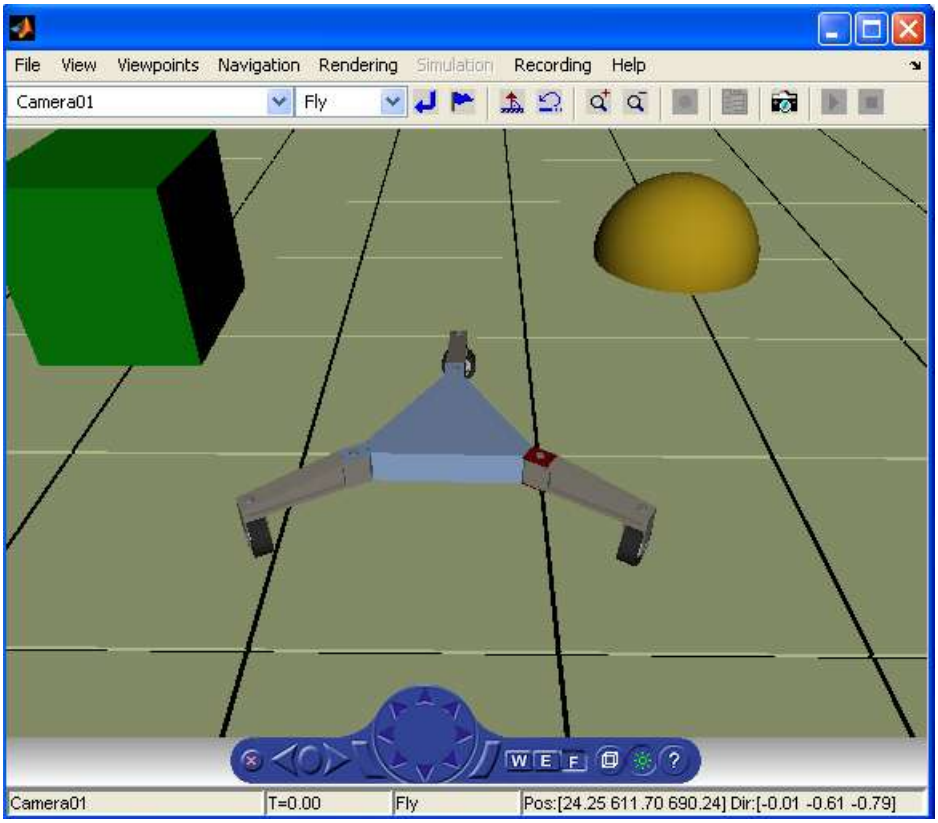


Figura 3.4.3. Mundo virtual del robot omnidireccional en Matlab.

Ahora es necesario examinar las propiedades del mundo virtual.

```
get(world);
```

Todos los elementos en el mundo virtual están representados como nodos VRML. El comportamiento puede ser controlado cambiando el campo en el nodo requerido. El comando NODES imprime en pantalla la lista de nodos disponibles en el mundo.

```
nodes(world)
```

Es necesario ahora crear todos los nodos VRML. El primer objeto nodo obtiene el nombre “tRobot”, el cual adquiere las propiedades del nodo “tRobot” del mundo virtual “world”. En el listado de abajo puede observarse todos los nodos que se manipulan en este proyecto.

```
tRobot = vrnode(world, 'tRobot');
leg01 = vrnode(world, 'leg01');
leg02 = vrnode(world, 'leg02');
leg03 = vrnode(world, 'leg03');
wheelbase01 = vrnode(world, 'wheelbase01');
wheelbase02 = vrnode(world, 'wheelbase02');
wheelbase03 = vrnode(world, 'wheelbase03');
wheel01 = vrnode(world, 'wheel01');
wheel02 = vrnode(world, 'wheel02');
wheel03 = vrnode(world, 'wheel03');
cam = vrnode(world,'Camera01');
```

En la parte siguiente del programa viene la definición de variables y constantes.

```
%definicion de constantes
k =180/pi;
%Velocidad inicial de las ruedas
w1x = -40;
w2x = -40;
w3x = -40;
%Definicion del radio de la rueda y longitud de las patas
R = 25;
L = 240;
%angulo inicial del robot
robotangle = 0;
factor = 1;
%tiempo del ciclo
t = 0.1;
```

Un punto importante en la simulación es obtener la posición global actual del robot, así como de la cámara que lo sigue y de los ángulos que mantienen las extremidades del robot.

```
%lee posicion actual del robot
sync(tRobot, 'translation', 'on');
tRobotPosition = tRobot.translation   %obtiene la position de la
llanta

%lee posicion actual de la camara
sync(cam, 'position', 'on');
camPosition = cam.position;   %get wheel position

%Rotacion inicial de las ruedas
wheelbase01.rotation = [0, 1, 0, 90/k];
wheelbase02.rotation = [0, 1, 0, 90/k];
wheelbase03.rotation = [0, 1, 0, 90/k];
```

Construcción de la matriz jacobiana correspondiente al modelo cinemático del robot.

```
%jacobiano del robot
J = [0, R/sqrt(3), -R/sqrt(3);
    (2*R)/3, -R/3, -R/3;
    -R/(3*L), -R/(3*L), -R/(3*L)];
```

Lo que viene a continuación es el esquema de un ciclo de control for para ejecutar varias veces el cálculo de la posición del robot, sumando el tiempo que se inicializa al principio y ejecutando el cálculo de la nueva posición del robot cada vez que se ejecuta un ciclo. De esta manera, se asigna nuevas posiciones globales al robot al vector v_c sumadas a las anteriores así como el ángulo de rotación.

Cálculo del nuevo ángulo del robot, se extrae del vector de posición v_c, en el elemento ω_c, y se suma a la variable robotangle.

```
%lectura del angulo actual del vector y asignacion a la variable
    %robotangle
    robotangle = robotangle + Vc(3) * t;

    %asignación del ángulo a la matriz de transformación del nodo
tRobot
    tRobot.rotation = [0, 1, 0, robotangle];
```

Cabe mencionar que se se realiza una suma cada ciclo tanto de las posiciones como del ángulo, debido a que el resultado de la multiplicación del jacobiano del robot y de las velocidades de las ruedas, se obtienen las velocidades en x y y del robot así como la angular, y debido a que se está ejecutando la simulación utilizando el modelo de cinemática directa, cada suma que se realiza se obtiene la posición global del robot así como su orientación.

La última parte del código muestra la suma y asignación de las nueva posición y orientación del robot, afectando directamente a la matriz de transformación de cada nodo.

```
%asignación de las coordenadas a la matriz de transformacion del nodo
    %de traslacion del robot.
    tRobotPosition(1) = ((Vc(1) * t) + tRobotPosition(1))/factor;
    tRobotPosition(3) = ((Vc(2) * t) + tRobotPosition(3))/factor;
    tRobot.translation = [tRobotPosition(1) tRobotPosition(2)
tRobotPosition(3)];

    %asignacion de coordenadas a la matriz de traslacion de la camara
    cam.position = [camPosition(1)+tRobotPosition(1)
camPosition(2)+tRobotPosition(2) camPosition(3)+tRobotPosition(3)];

    %actualizacion de la vista
    vrdrawnow;
    pause(t);
end
```

Simulación.

En la siguiente imagen se ve una secuencia de imágenes que ilustran parte de la simulación hecha con el modelo cinemático directo y el diseño del robot. Se toma en cuenta los parámetros originales en el código, que son las velocidades constantes de las ruedas a -40, para dar un ejemplo de movimiento circular, debido a que las ruedas disponen del mismo ángulo de rotación.

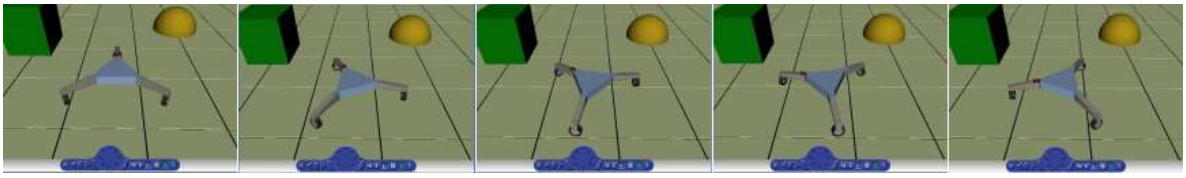


Figura 3.4.4. Secuencia del movimiento circular del robot.

El ejemplo de la Figura 3.4.5, es una demostración de un movimiento lineal del robot, es decir, que cambiando los parámetros de la velocidad de las ruedas es posible hacer que el robot se desplace en una línea recta, tal es el caso si modificamos las velocidades de la rueda 1 a 0, la rueda dos a 25 y la rueda 3 a -25. Con esta configuración hacemos que la rueda dos y tres giren en el sentido opuesto pero como están una de cada lado del robot, estas empujan en el mismo sentido, y la rueda 1 queda sin velocidad para evitar que el robot gire.

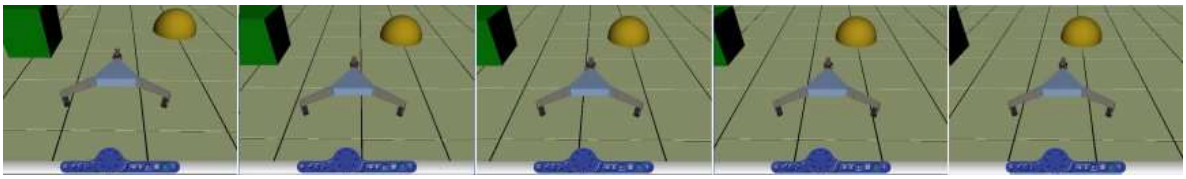


Figura 3.4.5. Secuencia del movimiento lineal del robot.

3.5. Modelado y simulación en Marilou.

En el transcurso del desarrollo del modelo del robot omnidireccional, se encontraron algunas herramientas que facilitan el desarrollo de aplicaciones de este tipo, además de acelerar de forma considerable el proceso de construcción de un simulador robótico. Incluso dan la facilidad de importar algunos objetos que pueden ser reusados y no necesariamente construidos dentro del entorno de estas aplicaciones. Tal es el caso del software Marilou, el cual es un sistema desarrollado explícitamente para la creación de mundos virtuales y la simulación de sistemas de robots, que no solo comprenden un solo robot sino la participación de mas de uno dentro del mismo entorno virtual, lo que permite entonces, ir mas allá en lo que se refiere a desarrollo de algoritmos inteligentes y de comportamiento de robots, ya que pudiera darse el caso de poder hacer que dos o mas robots interactúen entre ellos mismos en el mismo plano.

En el presente trabajo se lleva a cabo también la implementación del robot omnidireccional bajo el esquema que ofrece Marilou, esto con la finalidad de llegar a una evaluación final entre los métodos.

El proceso que se lleva a cabo para la construcción del modelo es de la siguiente manera:

- Preparación del proyecto.
- Construcción del robot.
- Construcción del mundo virtual.
- Verificación del proyecto.
- Programación.

Preparación del Proyecto.

Esta implementación comienza con la creación de un proyecto nuevo en Marilou, para esto es necesario seleccionar desde el menú “File” -> “New” -> “New Project”.

Una vez creado el proyecto es necesario hacer la configuración del mundo, el cual consiste en verificar los parámetros por defecto del sistema físico que se simulará, ya que pudiera darse el caso de poder simular bajo diferentes parámetros como una gravedad superior o inferior a la que físicamente es posible.

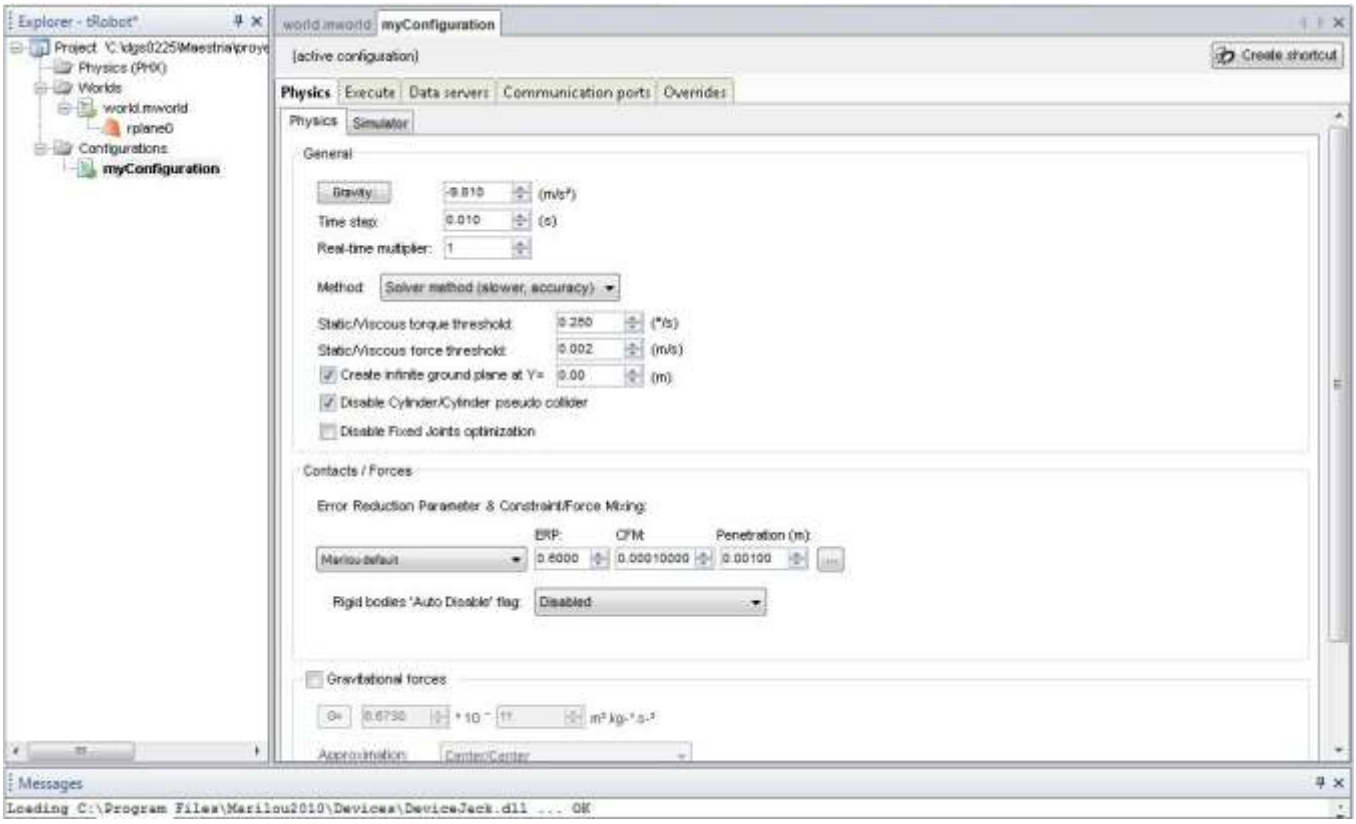


Figura 3.5.1. Diálogo de configuración del ambiente en Marilou.

Ya que se han revisado los parámetros, se puede dar inicio al diseño del robot. Creamos un físico desde el árbol que aparece a la izquierda del espacio de trabajo (ver Figura 3.5.1) bajo el nombre de “Physics (PHX)”, con el fin de crear nuestro primer robot. Desde esta opción haciendo clic derecho aparece un menú y desde allí agregar un nuevo physics con la opción “Add New PHX”, y se le da el nombre del robot que en este caso es “tRobot”.

Construcción del robot.

Ahora podemos comenzar a construir el modelo dentro de Marilou. El robot en Marilou se construye en tres fases:

- Partes dinámicas.
- Articulaciones y zonas.
- Dispositivos.

Partes dinámicas.

Las partes físicas se componen de todos los objetos geométricos que nos permitirán visualizar el comportamiento de forma gráfica y las propiedades dinámicas de cada objeto conocidas como cuerpos rígidos.

Como paso inicial en la construcción, importaremos el cuerpo triangular del robot creado en 3DStudio, seleccionando el físico “tRobot” se abren algunas opciones en la parte derecha del espacio de trabajo de Marilou (Figura 3.5.1).

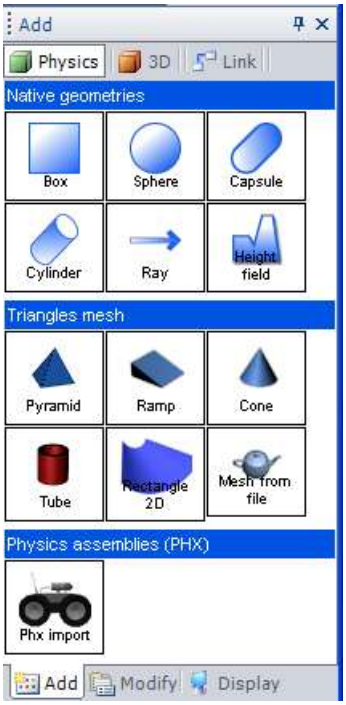


Figura 3.5.2. Objetos para construir el físico del robot.

El botón “Mesh from file”, permite hacer la importación de un objeto gráfico creado en otra aplicación como 3DStudio. Al Seleccionar esta opción se abre un diálogo en donde se puede buscar el archivo del objeto que se desea importar. Se abre el archivo “trianguloC.3ds” que contiene el dibujo del cuerpo triangular del robot y el formato está en 3ds de 3DStudio.

Una vez importado, se configuran las propiedades físicas como el peso que podría tener el objeto (parte derecha de la pantalla en la Figura 3.5.3), el cuerpo triangular se estima en 0.5 kgs. Las dimensiones del cuerpo son las que se dieron en 3DStudio, y se trasladan a milímetros por lo que el cuerpo del robot quedo con un diámetro de 120 milímetros debido a que se construyó a partir de un objeto Gengon.

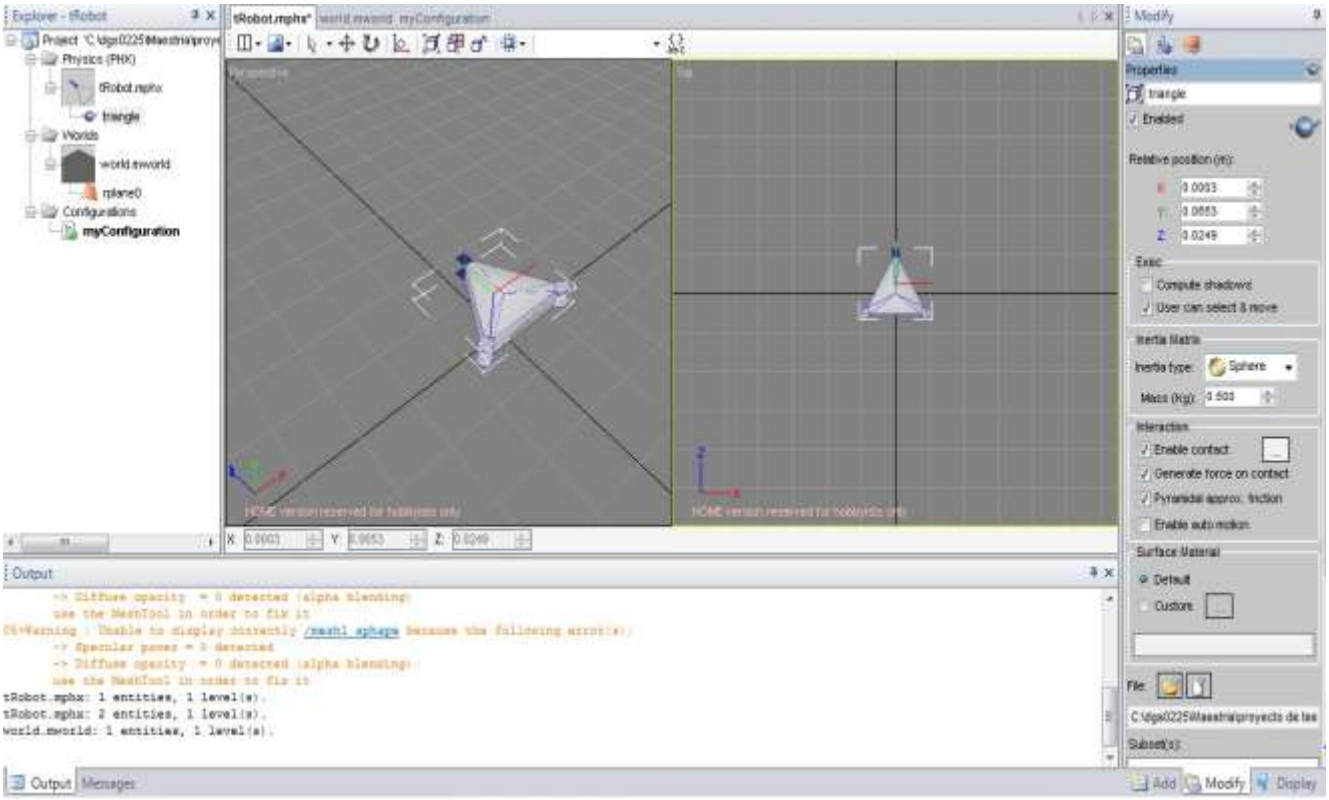


Figura 3.5.3. Cuerpo del robot y menú de propiedades del objeto.

En la columna de la derecha del espacio de trabajo de Marilou se pueden observar y modificar las propiedades físicas de cada objeto, por ejemplo, en el caso del cuerpo del robot, puede verse que tiene una posición relativa al mundo en x de 0.003, en y de 0.0653 y en z de 0.0249, que para nuestro caso no es relevante. Por otro lado, pueden asignarse propiedades físicas como de la masa y en todo caso puede seleccionarse una matriz de inercia de diferentes geometrías, estos datos participan en el cálculo de la dinámica del objeto.

Luego de haberse incluido el cuerpo del robot, se continua a agregar las extremidades. Marilou, permite también crear objetos básicos dentro del mismo entorno, así que no es necesario importar todos los objetos a razón de que sea necesario, ya que algunas figuras pueden no encontrarse dentro de la colección de objetos básicos que ofrece Marilou.

Para crear entonces la primera extremidad seleccionamos el botón “Box” en la pestaña “Add” (ver Figura 3.5.2.). Se puede dibujar el cuadro y enseguida darle las propiedades al objeto. Es importante definir las dimensiones del objeto y la masa estimada.

A la derecha se muestran los parámetros con los que se configura la primera extremidad. Se hace un estimado de peso de 0.1 kg y las dimensiones quedan en ancho de 150 milímetros y alto de 25 y ancho de 20.

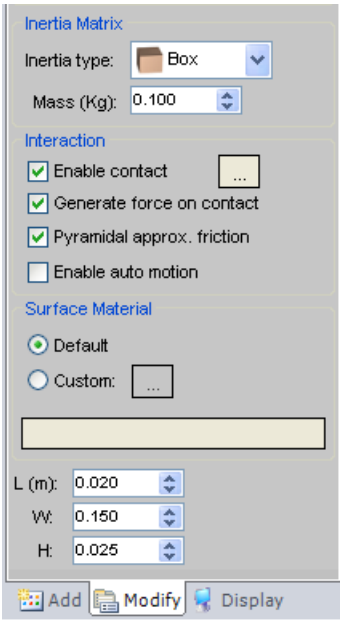


Figura 3.5.4. Parámetros de la pata del robot.

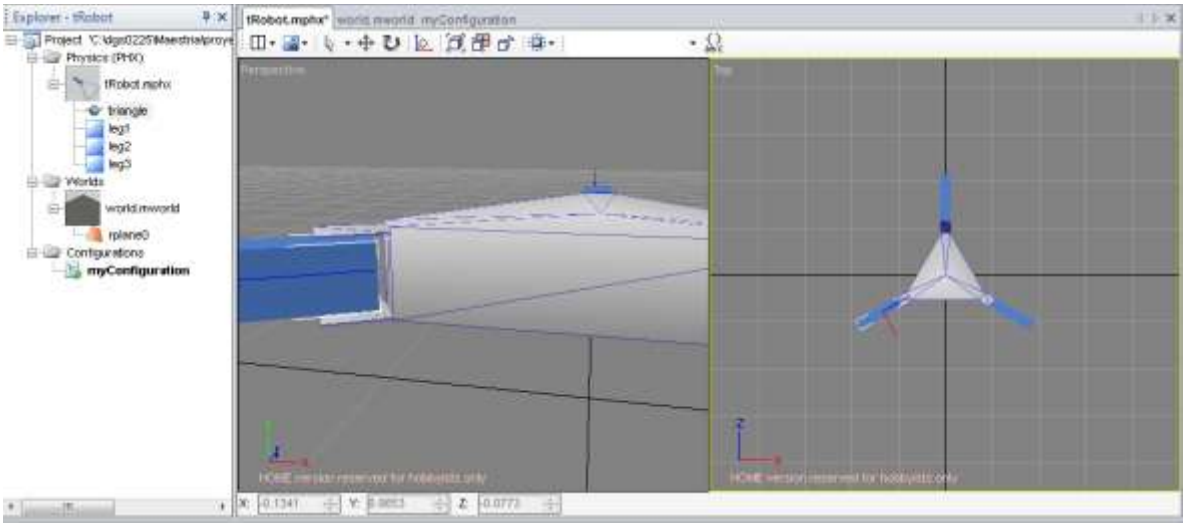


Figura 3.5.5. Cuerpo triangular y tres patas del robot.

El siguiente paso que se sigue es copiar la pata creada para generar las otras dos faltantes como se puede ver en la Figura 3.5.3.

Enseguida de haber creado las tres extremidades procedemos a la importación de las ruedas, el procedimiento que se sigue es similar al del cuerpo triangular, se agrega una malla “Mesh from file” y se selecciona el archivo “wheel2.3ds”. Se verifican las propiedades para una masa de 50 gramos y se hace copia para obtener tres ruedas, que se colocan en donde corresponde con cada pata (Figura 3.5.6).

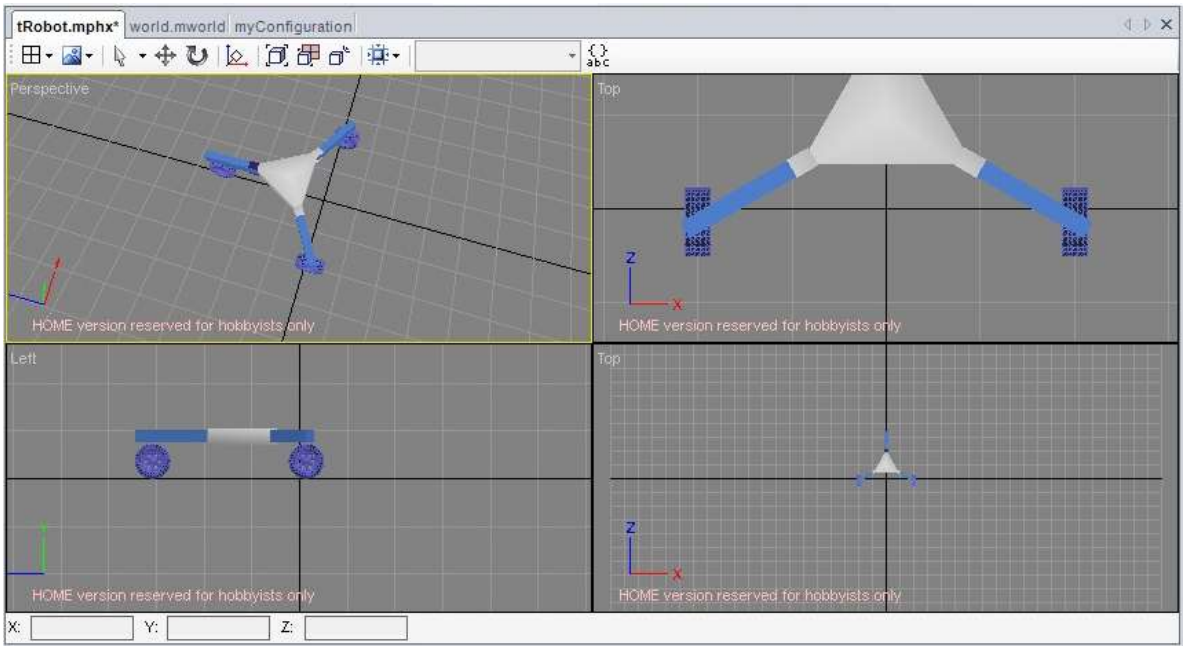


Figura 3.5.6. Diseño del robot omnidireccional completo en Marilou.

Hasta ahora solo se tienen algunos objetos geométricos a los cuales se les tienen que anexar propiedades dinámicas usando cuerpos rígidos, seleccionando todos los objetos con Ctrl + A, y haciendo clic en la botón “Rigid Body” . Seleccionar la opción “detached”. Aparecerán enmarcados los objetos con unos cuadros azules indicando que los objetos tienen ahora propiedades dinámicas, necesarias para la simulación de todo el entorno físico.

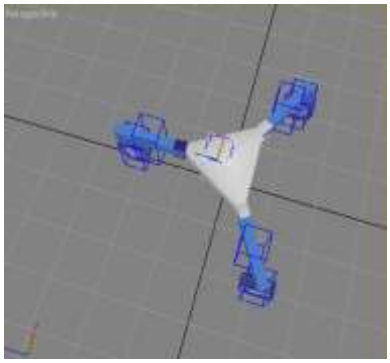


Figura 3.5.7. Asignación de cuerpo rígido a los objetos geométricos.

El marco de los cuerpos rígidos es puramente visual durante el diseño y desaparecen durante la simulación, aunque al ponerlos recientemente no aparecen uniformes y puede dificultar un poco el diseño, así que a manera de sugerencia se puede entrar a “Select mode” y seleccionar todos los cuerpos rígidos y en panel “Modify” darle el valor de 0.1 a “Anchor size”, este cambio se propaga a todos las entidades seleccionadas. Algo mas que puede hacerse es ubicar en otra posición a las entidades para facilitar aún más la selección de objetos en pantalla.

También, es necesario darle un nombre a cada objeto rígido para poder identificarlos en la parte donde se hace la unión entre ellos. Hasta ahora se han creado objetos geométricos que participan en la visualización y los objetos rígidos y se han denominado así:

Objeto gráifco	Objeto rígido	Descripción
triangle	bodyTriangle	Cuerpo del robot
leg1	bodyLeg1	Extremidad o pata del robot
leg2	bodyLeg2	Extremidad o pata del robot
leg3	bodyLeg3	Extremidad o pata del robot
wheel1	bodyWheel1	Rueda del robot
wheel2	bodyWheel2	Rueda del robot
wheel3	bodyWheel3	Rueda del robot

Tabla 3.5.1. Relación de cuerpos geométricos y rígidos.

Articulaciones y zonas.

Las articulaciones son los grados de libertad que se le quiere dar a cada objeto. Por ejemplo la rueda debe ser direccionable lo que significa que debe girar sobre su propio eje y además debe tener la habilidad de girar y crear movimiento de traslación, por lo que en este caso decimos que tiene dos grados de libertad. En Marilou esto se logra creando objetos de tipo “hinge” (bisagra) y asociándolas a los cuerpos rígidos que se han creado.

Comenzamos entonces creando un “hinge”. En la parte derecha del espacio de trabajo de Marilou, seleccionar la pestaña “add” en el fondo de la columna y seleccionar en la parte superior la opción “Link”.

Se escoge “Hinge” para la rueda, puede observarse que esta tiene dos ejes, uno horizontal que corresponde al giro de desplazamiento de la rueda y otro vertical que permite girar la rueda y dar dirección.

Se dibuja el objeto en el área de trabajo cerca de la rueda y se le da la dimensión aproximada arrastrando el mouse. Puede dársele un tamaño adecuado con la opción “Anchor size” en el panel “Modify”.

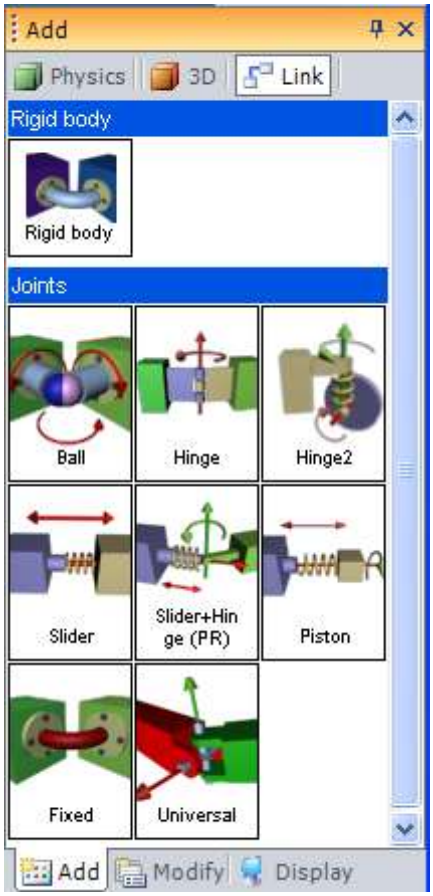


Figura 3.5.8. Panel de articulaciones.

Se le puede dar un nombre al objeto para poder identificarlo cuando se conecte a otro objeto, el nombre utilizado para la primer rueda es “hingeWheel1” y del eje de traslación “translationAxis” y “rotationAxis” para el eje de dirección. El objeto “hinge” se desplegará como un cuadro transparente con líneas en color rosa, el eje horizontal del “hinge2” es una flecha en color rojo y el vertical de dirección es en color verde. Se deben colocar en la posición exacta donde se desea dar movilidad al objeto (ver Figura 3.5.9).

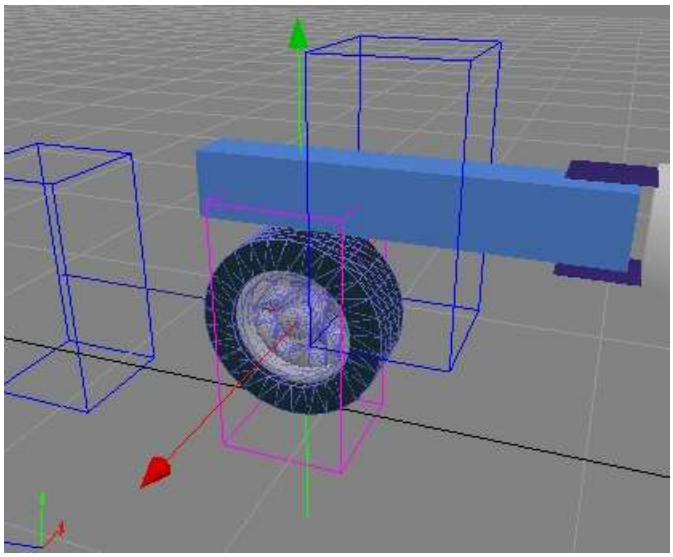


Figura 3.5.9. “Hinge2” colocado en la rueda.

Una vez que está en su lugar podemos entonces asociar el objeto “hinge2” a l cuerpo rígido, seleccionando el objeto “hinge2” y enseguida desde el panel “Modify”, seleccionar en la propiedad “Body1” el cuerpo rígido “bodyWheel1”, y en el “Body2” bodyLeg1”, es importante mantener este orden ya que el “Body1” se refiere al movimiento de traslación de la rueda y el “Body2” indica la conexión entre el eje de dirección de la rueda con la primera pata del robot.

Ahora se crea la siguiente articulación correspondiente a la conexión entre la pata y el cuerpo triangular del robot.

De vuelta se selecciona el panel “Add” y la opción “Link” (ver Figura 3.5.8) y se toma esta vez el “Hinge”. Se hace el mismo procedimiento arrastrando el mouse en la ventana de edición del robot y se asigna un tamaño de 0.1 desde el panel “Modify” y la casilla “Anchor size”. Se ubica el objeto en el eje que se desea controlar entre la pata y el cuerpo del robot. Es importante verificar la orientación de la flecha, esta debe quedar en el sentido de la bisagra que se desea articular, si no fuese así el objeto puede rotarse. Esta articulación se define con el nombre de “hingeLeg1” y su eje de dirección como “rotationAxis”.

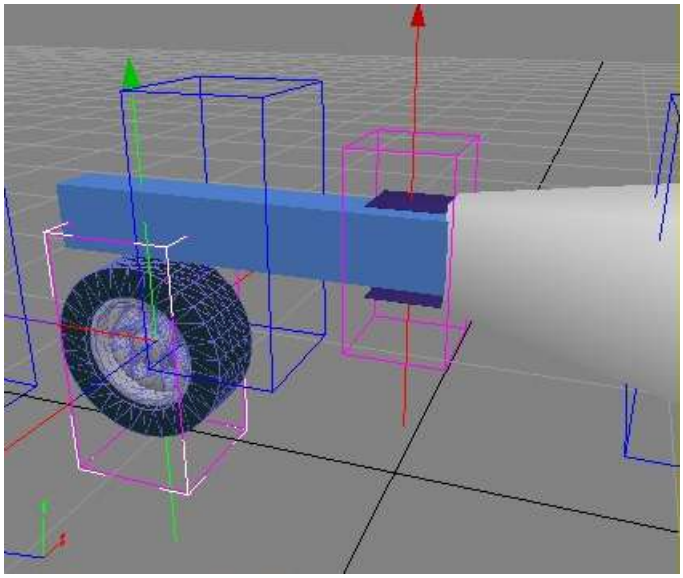


Figura 3.5.10. “Hinge” colocado entre la pata y el cuerpo del robot.

Para terminar esta conexión, se deben asociar los cuerpos rígidos. Se selecciona el objeto “Hinge” y desde el panel “Modify”, el “Body1” queda como “bodyLeg1” y la conexión con el “Body2” es “bodyTriangle”, que corresponde al cuerpo del robot (ver tabla 3.5.1).

Es necesario hacer esto para todas las patas y ruedas restantes del robot para quedar como se ilustra en la Figura 3.5.11.

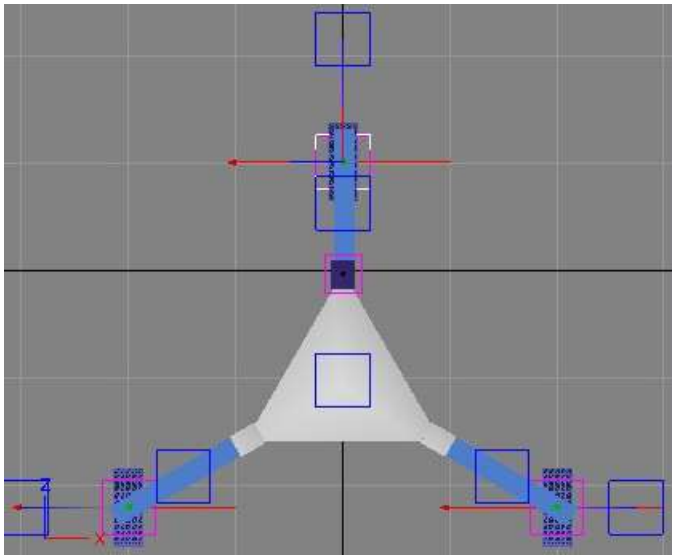
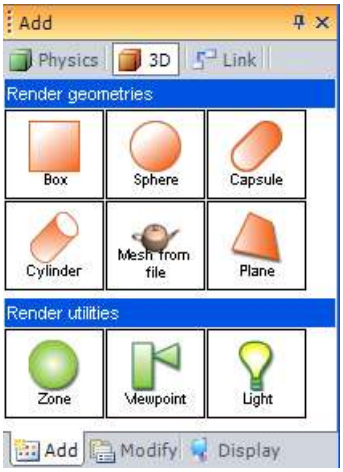


Figura 3.5.11. Cuerpos rígidos y articulaciones del robot.

Las siguientes partes son las zonas, estas definen el área de sensado, si es que se desea tener soporte de sensores de distancia, en algunas configuraciones como los robots móviles siempre es necesario contar con este tipo de dispositivos, para darle la funcionalidad de evitar colisiones Para el robot omnidireccional, se definen tres zonas de sensado, una en cada pata.

Para agregar un zona, desde el panel “Add”, y desde la opción en la parte superior “3D”, seleccionar el botón “Zone”.



Crear el objeto “Zone” al frente de la pata 1 del robot. Para hacer que esta zona de sensado se mueva con el robot también, se tiene que vincular a un cuerpo rígido. Para hacerlo se selecciona el objeto “Zone”, y presionando “Ctrl + L” se selecciona el cuerpo rígido de la pata.

Figura 3.5.12. Panel de 3D.

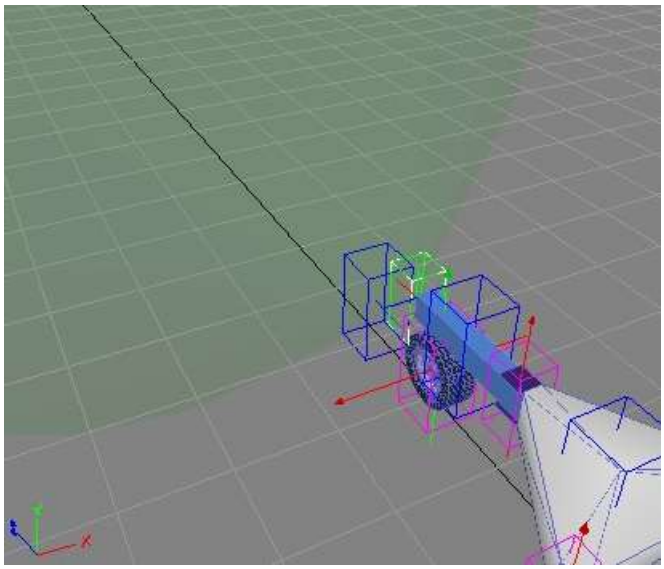


Figura 3.5.13. Zona de sensado de la pata 1.

Articulaciones	Tipo	Body1	Body2	Ejes
hingeWheel1	Hinge2	bodyWheel1	bodyLeg1	rotationAxis translationAxis
hingeWheel2	Hinge2	bodyWheel2	bodyLeg2	rotationAxis translationAxis
hingeWheel3	Hinge2	bodyWheel3	bodyLeg3	rotationAxis translationAxis
hingeLeg1	Hinge	bodyLeg1	bodyTriangle	rotationAxis
hingeLeg2	Hinge	bodyLeg2	bodyTriangle	rotationAxis
hingeLeg3	Hinge	bodyLeg3	bodyTriangle	rotationAxis
zoneLeg1	Zone	bodyLeg1		
zoneLeg2	Zone	bodyLeg2		
zoneLeg3	Zone	bodyLeg3		

Tabla 3.5.2. Relación de articulaciones contra cuerpos rígidos y ejes.

Dispositivos.

Ahora se vincularán los dispositivos al robot. Los dispositivos se asocian directamente a las entidades, como por ejemplo, un sensor de distancia se asocia a una zona, un motor se conecta a una articulación. Existen diferentes tipos de dispositivos disponibles en Marilou, hay desde motores simples, servomotores, cilindros actuadores, sensores de distancia, cámaras, etc.

Para agregar un dispositivo a la rueda seleccionar el eje horizontal "translationAxis" de la primer rueda, y se selecciona a la derecha el panel "Modify". En la parte superior hay botón "Devices", desde allí se puede seleccionar el tipo dispositivos que se quiere vincular con el botón  . , se selecciona "Motor" y nombrarlo como "motor".

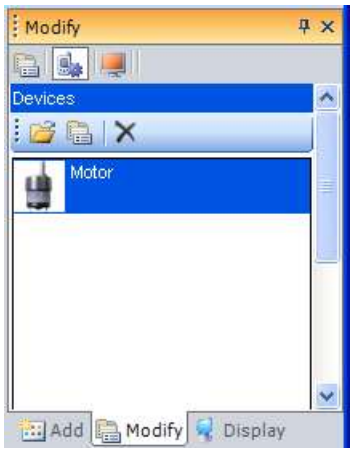


Figura 3.5.14. Panel Modify/Devices.

Enseguida se selecciona el eje "rotationAxis" de "hingeWheel1" y desde el panel "Modify" se selecciona "Devices" y agregar un "Servo", y se renombra como "servo". Se ejecuta la misma acción para el eje de rotación "rotationAxis" del hingeLeg1".

Todas las acciones se deben aplicar para las tres patas del robot. Ahora se agregan los sensores de distancia a cada zona, con el mismo procedimiento para los motores y servos, solo que ahora se selecciona el dispositivo "Distance" y se nombra como "sensor".

Construcción del mundo virtual.

El mundo virtual es donde el robot estará interactuando, este se puede crear tan complejo como se desee. Se pueden agregar objetos al mundo para obstaculizar el paso del robot y así observar los diferentes comportamientos que puede tomar en función del algoritmo implementado.

Para crear el mundo se realiza seleccionando el objeto “world.mworld“ desde el árbol “Worlds” a la izquierda de la pantalla. A dicho mundo se le agregan objetos como el plano que es el piso donde estarán todos los objetos, se agrega el robot, y así como una serie de objetos que son parte del mundo que se pretende simular.

Se inicia agregando el objeto “Plane” desde el panel “Add” y en objetos “3D”. Este plano queda centrado con una posición relativo en x=0, y=0, y z=0 y con una longitud y ancho de 10 metros.

Luego de aquí se anexan cuatro paredes para delimitar el área del robot a la longitud y ancho del plano. A las paredes se les dan dimensiones de 10 metros de largo por 0.5 de alto y 0.1 de ancho, y se disponen en todo el perímetro del plano. También, se les agrega la propiedad de cuerpo rígido con una masa de 100 kgs.

Se colocan un par de cajas con dimensiones distintas dispuestas dentro del plano y de las paredes del mundo, así como un objeto esférico para simular una pelota. Las dimensiones pueden ser variadas y en función de la necesidad de la simulación o del proyecto completo. Para este caso no es importante darles dimensiones precisas.

Finalmente, se incluye el objeto robot. Para ello, desde el panel “Add”, y en la opción “Physics” se selecciona el botón “Phx import”, aparece un diálogo donde

se puede seleccionar el robot que se ha creado previamente, el nombre asignado para el objeto robot dentro del mundo queda como tRobot1. Se pueden agregar más robots si fuera necesario.

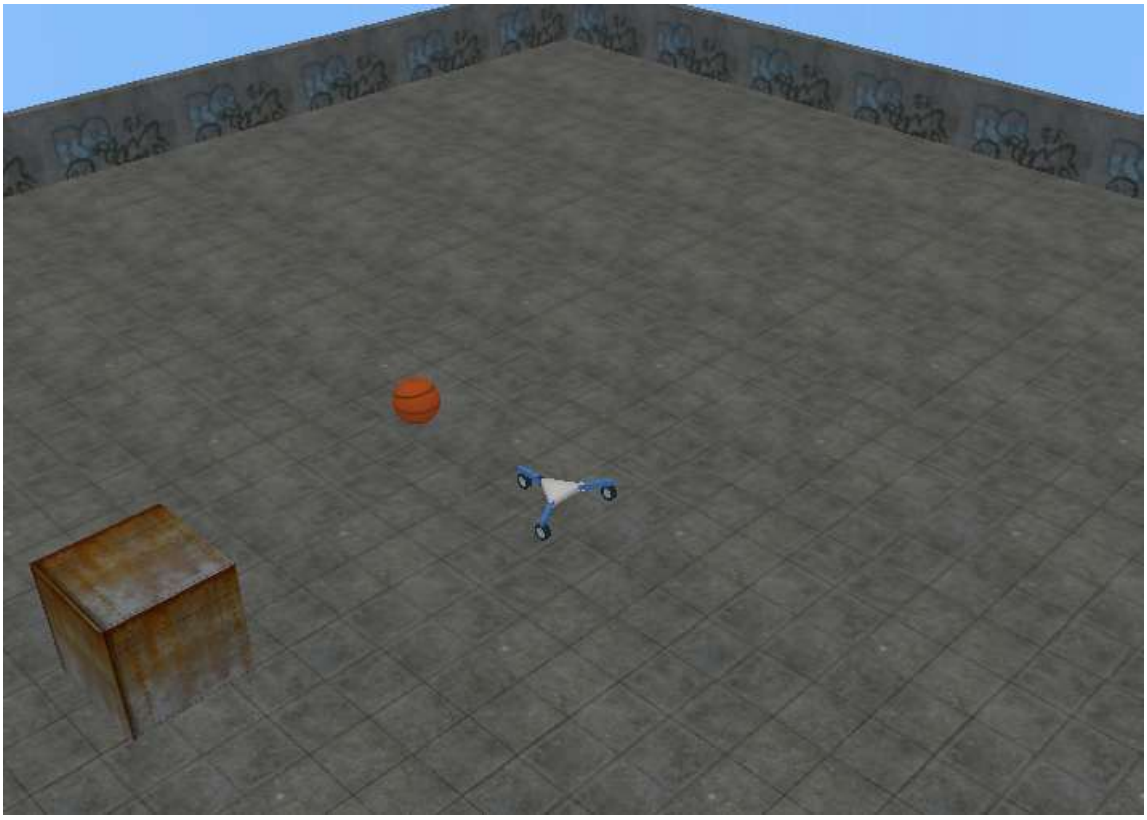


Figura 3.5.15. Mundo virtual y robot omnidireccional.

Verificación del proyecto.

Se realiza una verificación de todas las entidades recién construidas y para ver si los nombres están correctos. Este paso es importante, de esta manera se puede verificar que todo esté bien y listo para iniciar con la programación que es el siguiente paso.

Para verificar el estado del proyecto se corre la simulación haciendo clic en el botón  , y seleccione desde la opción “Tools” y “Devices Explorer”.

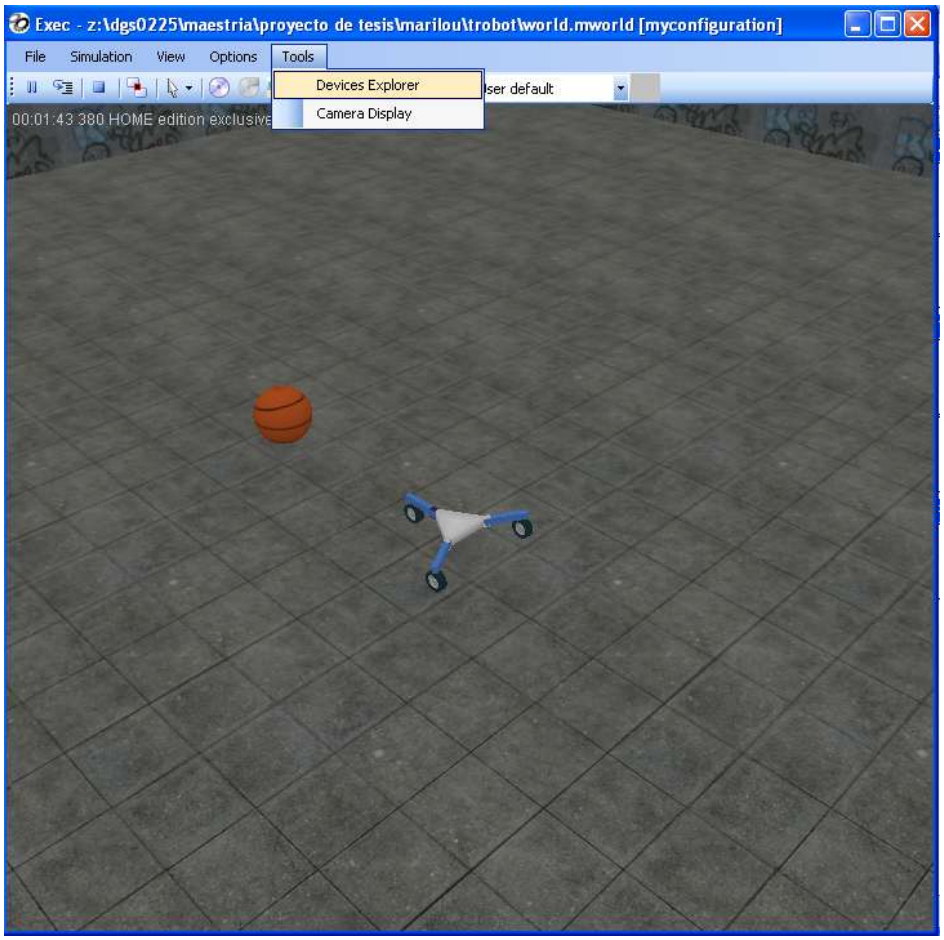


Figura 3.5.16. Simulación y verificación del proyecto.

En el diálogo de “Devices Explorer”, verificar que en el cuadro de texto esté localhost y luego clic en “Connect”. En el menú “Choose the RobotPHX” selecciones “\” para visualizar todos los dispositivos en el sistema y cotejar los nombres dados de alta.

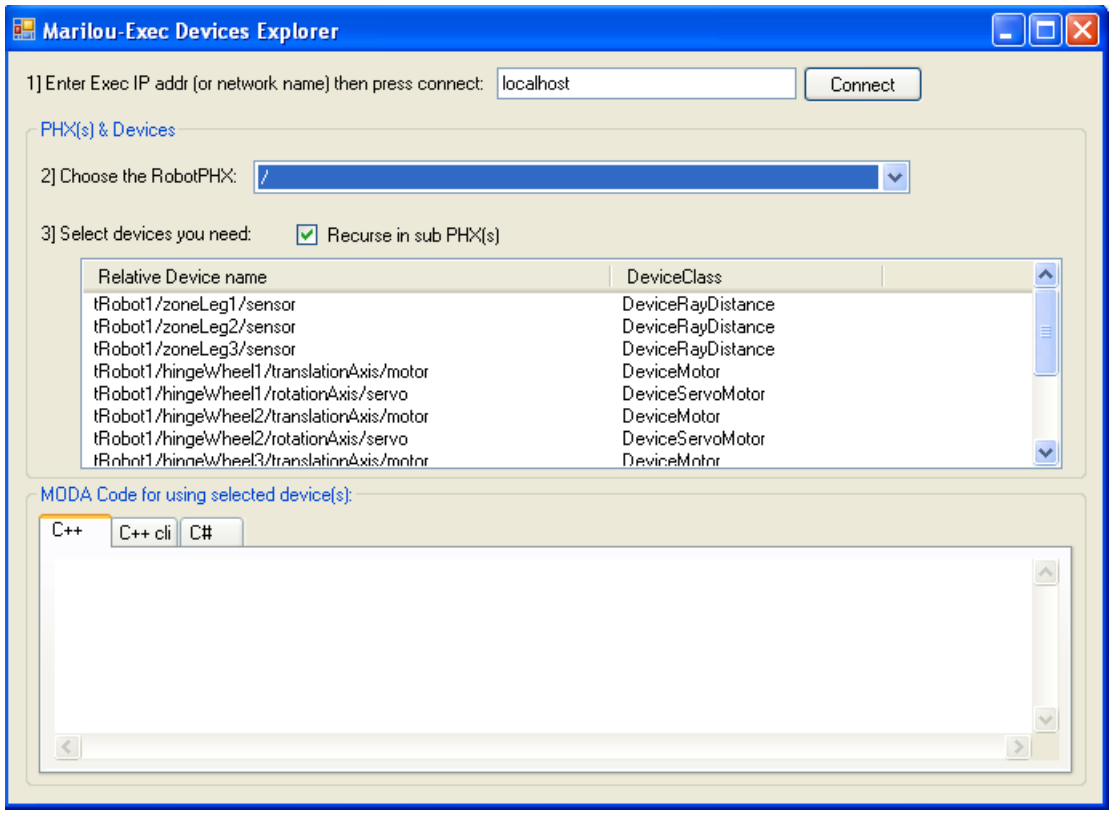


Figura 3.5.17. Diálogo “Device Explorer”.

Programación.

Marilou ofrece herramientas para la programación del control del robot. Tal es el caso que para el presente proyecto se construye una aplicación basada en el ambiente de trabajo de Microsoft Visual Studio 2005 C#.

Para crear la aplicación base, desde Marilou se selecciona el menú “File” luego “New” y “New from Wizards”, aparece un diálogo como el de la Figura 3.5.18. Desde allí se selecciona Visual Studio 2005 y “C# Forms Application”. Se crean una serie de archivos relacionados a la solución de Visual Studio 2005 C#.

Una vez creado el proyecto se puede abrir desde Visual Studio 2005, y se pueden editar las líneas para ajustarlo a las necesidades del proyecto. En el presente trabajo se construyó una aplicación simple para poder manipular manualmente los objetos construidos dentro del robot en Marilou (ver Figura 3.5.18). En el apéndice se incluye una copia impresa del código escrito en visual C# para esta aplicación.

Lo que la aplicación hace es básicamente conectarse al robot por medio de herramientas de red provistas por la Marilou, y crear dentro de visual C# los objetos que se crearon en Marilou, de tal suerte, que estos puedan manipularse y/o verificar el estado.

En la interface de control se incluyeron botones y deslizadores, para poder manipular los objetos como motores, por ejemplo, al hacer clic en el botón “Max” en el marco “Velocidad”, se le ordena al robot que arranque los motores a la máxima velocidad programada dentro de la aplicación en un sentido, si se selecciona “Min” se da la orden de arrancar los motores ahora en sentido opuesto.

Pero primero es necesario conectar la aplicación con el robot haciendo clic en la barra de estado al fondo del diálogo.

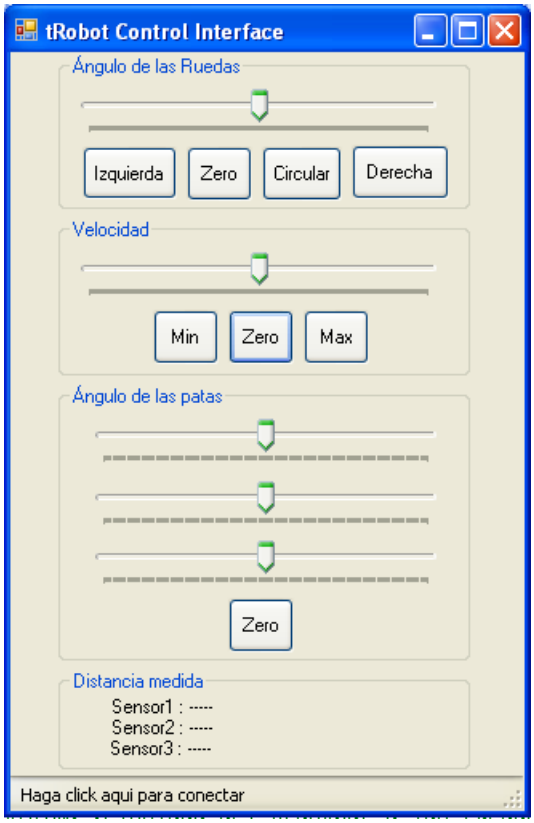


Figura 3.5.18. Interface de control del robot.

En el marco “Ángulo de las Ruedas”, se envía la orden de modificar la orientación de las ruedas para así darle dirección al robot, la aplicación envía este dato a las tres ruedas al mismo tiempo. En “Ángulo de las patas” se pueden modificar el ángulo de dirección de cada una de las patas por separado. Y al fondo de la interface se despliega el valor de la distancia de los tres sensores de distancia instalados en cada pata.

Pruebas del simulador.

Las pruebas que se pueden lograr bajo este esquema son muy básicas, aunque cabe mencionar que se pueden operar todos los movimientos permitidos por el robot, que para el caso son nueve grados de libertad, sin restricción. Todos los movimientos entonces son operados manualmente.

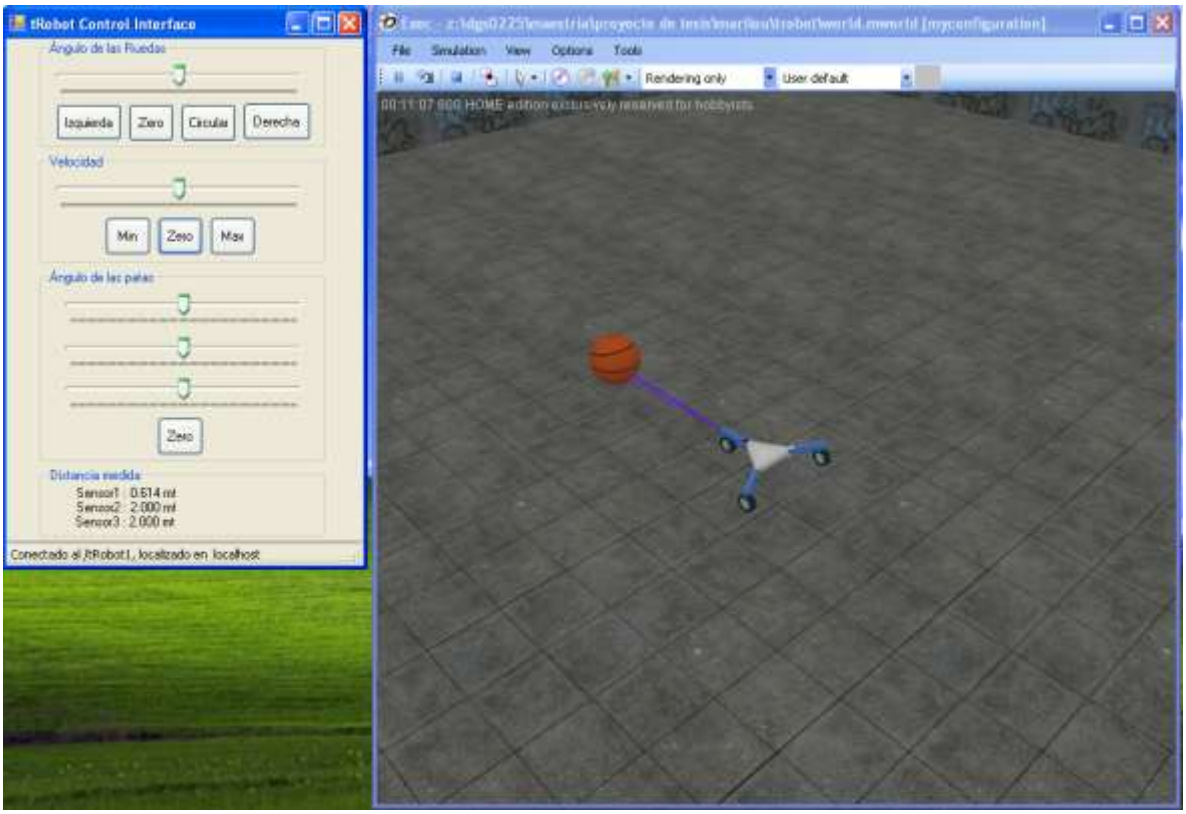


Figura 3.5.19. Conexión con el robot.

En el primer paso, que fue la conexión con la interface y el robot, se puede observar que el robot presenta mas vida, los sensores de distancia despliegan unas líneas indicando que están sensando y en la interface se puede observar que el sensor 1 muestra una distancia de 0.614 metros, mientras que en los otros dos despliegan 2.00 metros que es la distancia máxima de sensado permitida en esta aplicación.

En una primera prueba de movimiento, se selecciona una velocidad moderada con el deslizador en el cuadro de “Velocidad”, y se puede observar en la siguiente imagen como se ha desplazado el robot.

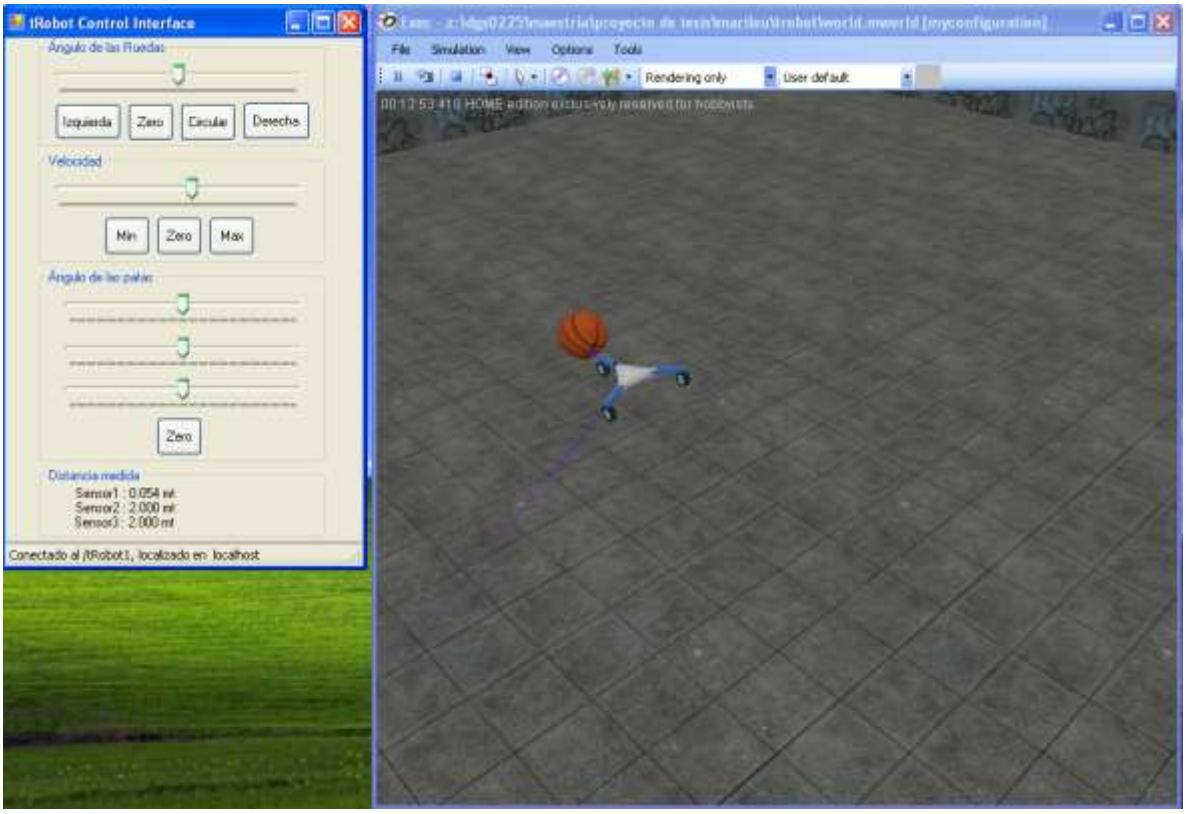


Figura 3.5.20. Prueba de movimiento del robot.

Se puede observar en la imagen 3.5.20 que el robot se ha desplazado y se ha acercado al balón con respecto a la imagen 3.5.19, desplegando incluso las líneas de sensado del sensor 2, en la distancia detectada en el sensor 1 se observa 0.054 mts.

La siguiente prueba se hace modificando el ángulo de las ruedas para modificar la dirección del robot, y el robot después de un tiempo se observa que se ha desplazado hacia otro espacio dentro del plano (ver Figura 3.5.21).

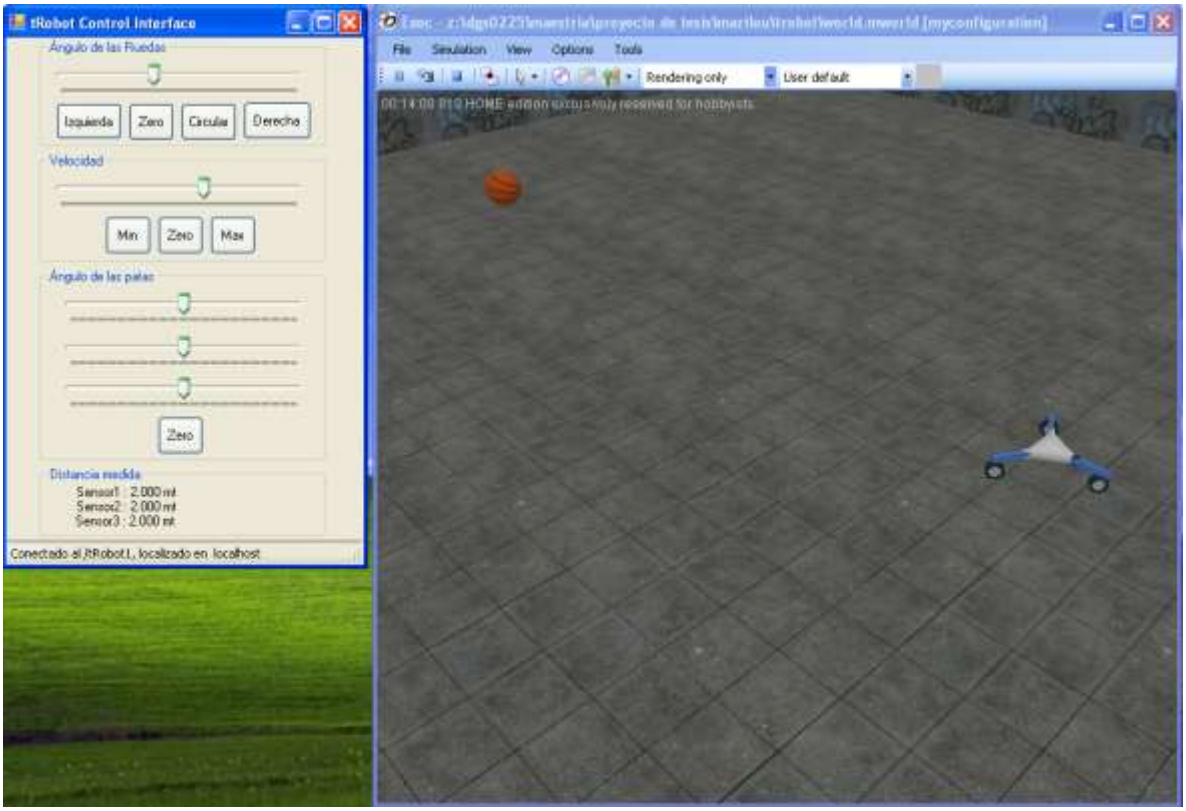


Figura 3.5.21. Prueba de dirección del robot.

En esta prueba se puede ver que los sensores despliegan 2.000 metros indicando que no se encuentra ni cerca de las paredes que delimitan el plano creado en el mundo para esta aplicación.

En la última prueba puede observarse como se mueve la rotación de las patas y de las ruedas en la dirección que se desee.

De esta manera, se demuestra la completa funcionalidad del modelo del robot omnidireccional creado en Marilou.

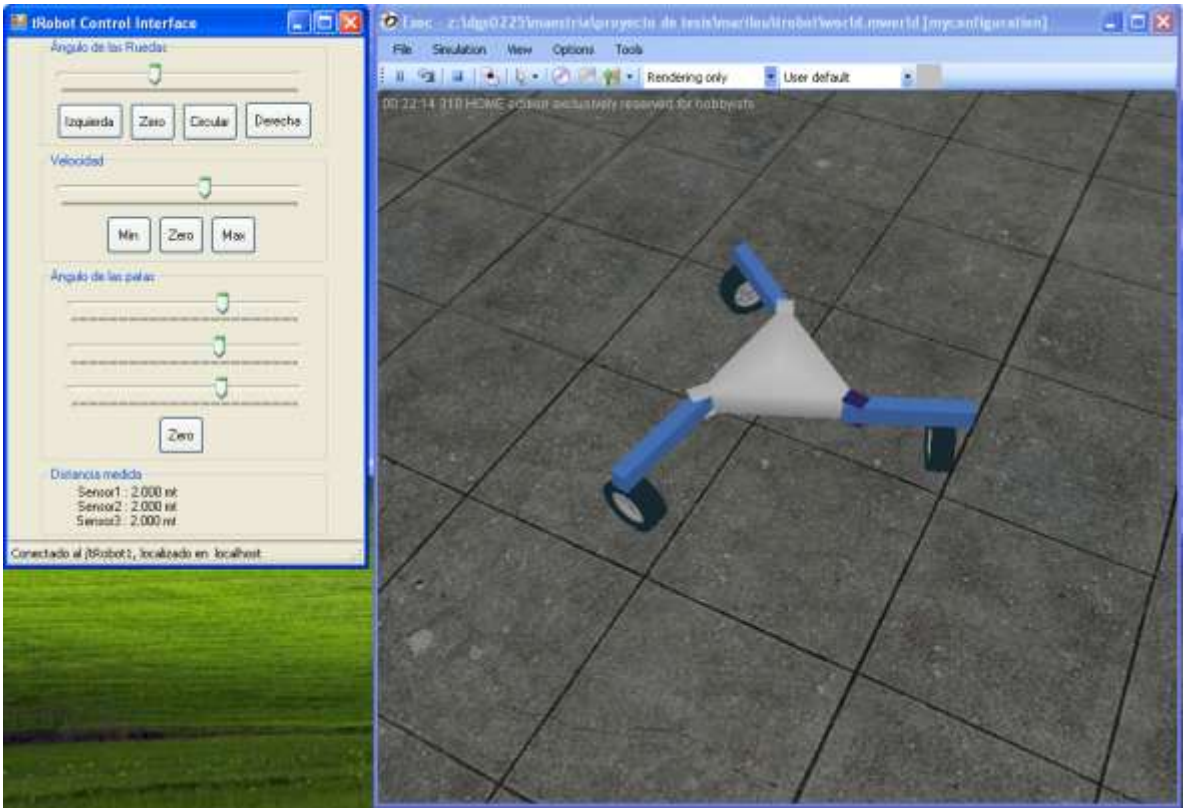


Figura 3.5.22. Prueba de rotación de las patas y las ruedas.

4. RESULTADOS Y DISCUSION

4.1. Resultados Generales.

En general los resultados que se obtuvieron fueron satisfactorios y por otro lado el proyecto aporto mucho mas que solo un buen resultado a nuestro entendimiento sobre los temas de robótica y particularmente hablando de la cinemática, que ahora pasa a ser un tema de mucho más interés y que permitirá llegar a desarrollar trabajos en la misma área en un futuro.

Se considera que los resultados se ilustran durante el desarrollo de la metodología, por mencionar alguno, el resultado obtenido de la simulación en Matlab (Sección 3.4). Aunque el desarrollo del código no es tan extenso, es debido a que atrás de todo esto se encuentra también un desarrollo del modelo cinemático del robot en cuestión, y por otro lado, el desarrollo del diseño del robot omnidireccional con todas sus piezas en 3D Studio.

En el caso del uso del software Marilou, presenta una menor dificultad debido a que esta es una aplicación construida expresamente para el desarrollo de simuladores de robots, y que además, incluye características como las de las propiedades dinámicas de los objetos, lo que agrega más realismo a la simulación a tal punto que cuando se está manipulando el robot, debe tenerse cuidado en restringir los grados de actuación que se puede dejar, debido a que el robot puede incluso caerse como en un mundo completamente real, esto al rotar las patas todas en el mismo sentido, el robot pierde equilibrio. Del mismo modo, en Marilou se tiene la capacidad de modificar las características físicas del plano, de tal manera que pudiera simularse un piso rugoso o tal vez uno deslizante. En algunos

casos pudiera darse la necesidad de crear superficies irregulares como para observar las propiedades de escalabilidad del robot.

Tal vez, realizando una comparación entre ambos métodos, es que el primero requiere de un gran esfuerzo en la obtención del modelo cinemático del robot, además de realizar verificaciones al modelo final del robot para determinar si existen errores en los cálculos o no, aunque por otro lado, una vez obtenido el modelo y que se haya verificado su asertividad, el modelo puede ser fácilmente insertado dentro del código de programación de Matlab, y manipularlo con algoritmos de inteligencia artificial como lógica difusa o redes neuronales, de lo cual Matlab cuenta con una basta librería.

Si se asoma uno al método de diseño con Marilou, este al principio puede parecer de fácil uso y donde se pueden desarrollar modelos de robots que ya incluyen implícitamente su cálculo cinemático y además el dinámico, al final, no se dispone de una herramienta en donde se puedan manipular los grados de acción del robot de forma automática como lo es un algoritmo de control. Entonces, para ello es necesario que se escriba un código adicional dedicado a dicho robot.

4.2. Trabajo Futuro.

En el caso de la metodología implementada con el uso de Matlab, sería interesante el poder aplicar este modelo completo del robot en alguna tarea específica. Por ejemplo, una de las intenciones del desarrollo de simuladores de robots o incluso de la construcción física es la aplicación en alguna investigación particular, tal es el caso, de desarrollar algoritmos que controlen de forma automática cierto comportamiento que se desee. Otra forma de aplicarlo, es de forma didáctica, en donde, se usen aplicaciones como Matlab y se desarrollen modelos de inteligencia artificial para manipular el robot sin tener que emplear mas tiempo en el desarrollo del prototipo. Algunas de estas técnicas apuntan a la

utilización de técnicas de inteligencia artificial para la generación y control de trayectorias de robots móviles, por mencionar alguna está el aprendizaje por medio de redes neuronales o el control para solucionar problemas de trayectorias (Nardenio A. Martis et al 2008).

Específicamente hablando del desarrollo en Marilou se puede pensar en que la metodología apoye en la construcción de otros robots con otras características y morfología, y tal vez desarrollar un módulo que permita conectarlo con Matlab, de esta forma se obtiene lo mejor de ambos métodos que es la velocidad de creación de un modelo robótico que incluya propiedades cinemáticas y dinámicas en el modelo, y de la utilización de librerías de inteligencia artificial de Matlab. Bajo este esquema, se puede pensar en trabajar en una librería que permita comunicar ambas aplicaciones y así desarrollar más modelos a una mayor velocidad. Otra de las tendencias que se sigue es la colaboración entre robots (Li Xiang, et al 2008), en donde un grupo de robots forman un equipo para realizar tareas en conjunto, tal es el caso de los eventos de concurso de futbol donde participan equipos de robots móviles, entonces para llevar a cabo una simulación de varios robots al mismo tiempo y ejecutar un algoritmo para cada uno de ellos es una tarea que demanda mucho tiempo y esfuerzo y es allí donde entran este tipo de herramientas. Por otro lado, los robots multi-podos es otra de las formas de estudio de los robots, los cuales permiten reproducir las habilidades que algunos animales tales como insectos (García-López, M.C. et al. 2012), lo cual permite que el robot acceda a lugares con superficies más caprichosas que para un robot con ruedas le sea difícil explorar.

5. REFERENCIAS BIBLIOGRÁFICAS

Bravo Hernández Paola, et al. 2009. *Diseño de un Simulador 3D para Robots Manipuladores de 3 Grados de Libertad. Misión Mecatrónica. Número 1, Año 4,2010.* México.

García-López, M.C.*, Gorrostieta-Hurtado, E.a. Vargas-Soto, E.a, Ramos-Arreguín, J.M.a, Sotomayor-Olmedo, A.a, Moya Morales, J.C.a. 2012. *Kinematic analysis for trajectory generation in one leg of a hexapod robot..* Facultad de Informática-Universidad Autónoma de Querétaro.México.

He Bin, Liu Wen Zhen, Lv Hai Feng. 2010. *The Kinematics Model of a Two-wheeled Self-balancing Autonomous Mobile Robot and Its Simulation*. Shangai University.

Jae-Bok Song*, Kyoung-Seok Byun**. 2008. *Design and Control of an Omnidirectional Mobile Robot with Steerable Omnidirectional Wheels*. *Korea University, **Mokpo National University. Republic of Korea. P223.

Li Xiang, Li Xunbo and Chen Liang. 2008. *Multi-disciplinary Modeling and Colaborative Simulation of Multi-robots System Based on HLA.* University of Electronic Science and Technology of China, Chegdu. China.

Masato Ishikawa. 2010. *Development and Control Experiment of the Trident Snake Robot*. IEEE 2010.

Michael Todd Peterson. 2000. *3D Studio MAX 3*. Prentice Hall.

Moore, Holly. 2007. *MATLAB® para ingenieros*. Prentice Hall.

Nardenio A. Martins. Douglas Bertol, Warody C. Lombardi, Edson R. Pieri, María M. Dias. 2008. *Neural Control Applied to the Problem of Trajectory Tracking of Mobile Robots with Uncertainties*. Universidade Fderal de Santa Catarinal. Maringá, Paraná, Brasil.

Nory Afzan Mohd Johari, Habibollah Haron, Abdul Syukor Mohamad Jaya. 2007. *Robotic modeling and simulation of palletizer robot using Workspace5*. Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.

Ollero, Anibal 2001. *Robótica Manipuladores y Robots Móviles.* Marcombo. Barcelona, España.

Romero Torres Ruben*, Gorrostieta Hurtado Efrén**, Ramos Arreguin Juan Manuel**, Pedraza Ortega Carlos**, González Gutiérrez Carlos Alberto*, González Aguirre Marco Antonio*, Villaseñor Carrillo Ubaldo Giovanni*, Collazo Cuervas José Iván*. 2009. *Diseño Mecatrónico de un Robot Omni-direccional de Nueve Grados de Libertad*. *Universidad del Valle de México y **Universidad Autónoma de Querétaro.

Rui Ding, Junzhi Yu, Qinghai Yang, and Min Tan. 2009. *Kinematics Modeling and Simulation for an Amphibious Robot: Design and Implementation*. Institute of Automation, Chinese Academy of Sciences. Beijing 100190, China.

Teerawat Thepmanee, et al. 2007. *A Simple Technique to Modeling and Simulation Four Axe Robot-Arm Control.* Department of Instrumentation Engineering. Thailand.

V.F. Muñoz Martinez, G. Gil-Gómez y A. García Cerezo 2010. *Modelado Cinemático y Dinámico de un Robot Móvil Omni-Direccional*. Universidad de Málaga. Parque Tecnológico de Andalucía C/Severo Ochoa 4, 20590 Málaga.

Yuang Shaoqiang, Liu Zhong, Li Xingshan. 2008. Modeling and Simulation of Robot Based on Matlab/SimMechanics. Beijing 100083, P.R. China.

APENDICE

A. 1. Código Fuente de al Interface de Control en Visual C #.

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using Moda;

namespace tRobot
      {
      public partial class Form1 : Form
            {
        bool _robotFound;
            private Moda.Connection _Connection;
            private Moda.RobotPHX _MyRobot;

        private Moda.DeviceMotor[] _MyWheelMotor = new
Moda.DeviceMotor[4];
        private Moda.DeviceServoMotor[] _MyWheelServo = new
Moda.DeviceServoMotor[4];
        private Moda.DeviceServoMotor[] _MyLegServo = new
Moda.DeviceServoMotor[4];
        private Moda.DeviceDistance[] _MySensor = new
Moda.DeviceDistance[4];

            private const String ModaServerAddress ="localhost"; //el
puerto usado para la conexión es 13000
            private const String RobotName = "/tRobot1";


            public Form1()
                  {
                  //Inits
            _robotFound = false;
                  _Connection = null;
                  _MyRobot = null;



                  //Very usefull to debug application
                  AllocConsole();

                  InitializeComponent();

            //avisa en la barra de estado que se haga la conexión con el
robot
            toolStripStatusLabel1.Text = " Haga click aqui para conectar
";

            //Valores iniciales de las etiquetas de despliegue de
distancia de sensores
```

```
            label1.Text = "Sensor1 : -----";
            label2.Text = "Sensor2 : -----";
            label3.Text = "Sensor3 : -----";

            //Habilita al timer de actualización de datos
            timer1.Enabled = true;

                  }

          private bool InitConnection(String ServerIP,String RobotName)
        {
                  bool bRet = false;

                  //Connect to MODA server
                  if(_Connection.Connect(ServerIP))
                        {
                        Console.WriteLine("Connected to " + ServerIP);


                        //Find the robot
                        _MyRobot=_Connection.QueryRobotPHX(RobotName);
                        if(_MyRobot!=null)
                              {
                                  Console.WriteLine("Robot "+RobotName + "
found");

                                  bRet=true;
                              }
                        else
                              {
                                  Console.WriteLine("Cannot find robot
"+RobotName);

                         //si no existe el nombre del robot, despliega un
mensaje en pantalla
                         MessageBox.Show("No se encontró el robot " +
RobotName);
                              }
                        }
                  else
                        {
                            Console.WriteLine("Cannot connect to
"+ServerIP);

                    //si no existe el servidor despliega el mensaje con
el nombre del servidor o la dirección IP
                      MessageBox.Show("No se pudo conectar a " + ServerIP);
                        }
                  return(bRet);
            }

            [DllImport("KERNEL32.DLL", EntryPoint="AllocConsole")]
            public static extern bool AllocConsole();

        private void Form1_Load(object sender, EventArgs e)
        {
```

```
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            if (_robotFound)
            {
                label1.Text = "Sensor1 : " +
_MySensor[1].GetMeasure().ToString("0.000 mt");
                label2.Text = "Sensor2 : " +
_MySensor[2].GetMeasure().ToString("0.000 mt");
                label3.Text = "Sensor3 : " +
_MySensor[3].GetMeasure().ToString("0.000 mt");
            }
        }

        private void toolStripStatusLabel1_Click(object sender, EventArgs
e)
        {

            //Despliega texto
            toolStripStatusLabel1.Text = " Conectando...";

            //actualiza lo que está en la interface
            this.Update();

            //Crea el objeto de conexion de la clase MODA.
            _Connection = new Moda.Connection(true);

            //intenta conectar al servidor
            //Conecta con el robot, haciendo la llamada a la función
InitConnection
            //Se da el argumento con la dirección del servidor donde
reside el robot
            //Y se da el nombre del robot que reside en dicho servidor
            if (InitConnection(ModaServerAddress, RobotName))
            {
                toolStripStatusLabel1.Text = "Conectado al " + RobotName
+ ", localizado en  " + ModaServerAddress;
                _robotFound = true;

                //si se conectó con el robot entonces conectar con los
dispositivos del robot
                for (int i = 1; i < 4; i++)
                {
                    _MyLegServo[i] =
_MyRobot.QueryDeviceServoMotor("hingeLeg" + i.ToString() +
"/rotationAxis/servo");
                    _MyWheelServo[i] =
_MyRobot.QueryDeviceServoMotor("hingeWheel" + i.ToString() +
"/rotationAxis/servo");
                    _MyWheelMotor[i] =
_MyRobot.QueryDeviceMotor("hingeWheel" + i.ToString() +
"/translationAxis/motor");
                    _MySensor[i] = _MyRobot.QueryDeviceDistance("zoneLeg"
+ i.ToString() + "/sensor");

                    _MyLegServo[i].SetTorque(2);
```

```
                    _MyLegServo[i].SetVelocityDPS(100);
                    _MyLegServo[i].GoPositionDeg(0);

                    _MyWheelServo[i].SetTorque(2);
                    _MyWheelServo[i].SetVelocityDPS(100);
                    _MyWheelServo[i].GoPositionDeg(0);

                    _MyWheelMotor[i].SetTorque(2);
                    _MyWheelMotor[i].SetVelocityDPS(0);

                }
            }
            else
            {
                _Connection = null;
                _MyRobot = null;
                _robotFound = false;
                toolStripStatusLabel1.Text = "No se pudo conectar al
robot " + RobotName;
            }
        }

        //Control del angulo de las ruedas
        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            if (_robotFound)
            {
                for (int i = 1; i < 4; i++)
                {

this._MyWheelServo[i].GoPositionDeg((float)trackBar1.Value);
                }
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        //Control velocidad de las ruedas
        private void trackBar2_Scroll(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                //mueve el valor del trackbar2 a la velocidad de las
ruedas
                for (int i = 1; i < 4; i++)
                {

_MyWheelMotor[i].SetVelocityDPS((float)trackBar2.Value);
                }
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }
```

```
        //Regresa a Cero la velocida de todas las ruedas
        private void button1_Click(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                for (int i = 1; i < 4; i++)
                {
                    _MyWheelMotor[i].SetVelocityDPS(0);
                }
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        //Regresa a cero todos los angulos de los servos de las ruedas
        private void button4_Click(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                for (int i = 1; i < 4; i++)
                {
                    this._MyWheelServo[i].GoPositionDeg(0);
                }
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        //pone maxima velocidad al movimiento de traslación de las ruedas
        private void button2_Click(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                for (int i = 1; i < 4; i++)
                {

_MyWheelMotor[i].SetVelocityDPS((float)trackBar2.Maximum);
                }
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        //mueve todas las ruedas para que gire en circulos
        private void button5_Click(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                _MyWheelServo[1].GoPositionDeg(90);
                _MyWheelServo[2].GoPositionDeg(210);
```

```
                _MyWheelServo[3].GoPositionDeg(-30);
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        //Gira todas las ruedas para voltear a la derecha
        private void button6_Click(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                _MyWheelServo[1].GoPositionDeg(90);
                _MyWheelServo[2].GoPositionDeg(90);
                _MyWheelServo[3].GoPositionDeg(90);
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        //Gira todas las ruedas para voltear a la izquierda
        private void button7_Click(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                _MyWheelServo[1].GoPositionDeg(-90);
                _MyWheelServo[2].GoPositionDeg(-90);
                _MyWheelServo[3].GoPositionDeg(-90);
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        //pone la mínima velocidad del trackbar a las ruedas
        private void button3_Click(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                for (int i = 1; i < 4; i++)
                {

_MyWheelMotor[i].SetVelocityDPS((float)trackBar2.Minimum);
                }
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        //controla la posición de la pata 1
        private void trackBar3_Scroll(object sender, EventArgs e)
```

```
        {
            if (_robotFound)    //si se está conectado al robot
            {
                _MyLegServo[1].GoPositionDeg((float) trackBar3.Value);
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        private void trackBar4_Scroll(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                _MyLegServo[2].GoPositionDeg((float)trackBar4.Value);
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        private void trackBar5_Scroll(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                _MyLegServo[3].GoPositionDeg((float)trackBar5.Value);
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }

        //Regresa el angulo de las patas a cero
        private void button8_Click(object sender, EventArgs e)
        {
            if (_robotFound)    //si se está conectado al robot
            {
                for (int i = 1; i < 4; i++)
                {
                    this._MyLegServo[i].GoPositionDeg(0);
                }
            }
            else
            {
                MessageBox.Show("No hay conexión con el robot");
            }
        }
    }
                                    }
```